Session and Terminal Mobility: Approaches and Experiments

Fernando Pedro Moreira
## Contents

1 Introduction
   1.1 Evolution of mobile communications ........................................... 4
   1.2 Objectives .............................................................................. 4
   1.3 Expected Results ....................................................................... 5
   1.4 Document Structure .................................................................. 5

2 State of the Art
   2.1 Mobile IP - MIP ........................................................................ 7
      2.1.1 Introduction ....................................................................... 7
      2.1.2 Terminology ....................................................................... 7
      2.1.3 Basic Operation ................................................................... 8
      2.1.4 Security ............................................................................. 9
   2.2 Host Identity Protocol - HIP ........................................................... 11
      2.2.1 Introduction ....................................................................... 11
      2.2.2 Terminology ....................................................................... 11
      2.2.3 Host Identifier - HI ................................................................. 11
      2.2.4 Basic Operation ................................................................... 12
      2.2.5 Conclusions ....................................................................... 16
   2.3 Session Initiation Protocol - SIP ...................................................... 17
      2.3.1 Introduction ....................................................................... 17
      2.3.2 Terminology ....................................................................... 17
      2.3.3 SIP and SIPS URLs ................................................................. 18
      2.3.4 SIP's REFER Extension ........................................................... 18
      2.3.5 Operation ............................................................................ 19
      2.3.6 Mobility ............................................................................. 23
   2.4 State of the Art Conclusions ............................................................. 25

3 Work Description
   3.1 HIP Test bed ............................................................................ 27
      3.1.1 Objectives ......................................................................... 27
      3.1.2 Motivation .......................................................................... 27
      3.1.3 Basic Test-bed ..................................................................... 27
      3.1.4 Test-bed with RVS ................................................................. 29
      3.1.5 HIPL .................................................................................. 30
      3.1.6 HIP4BSD .......................................................................... 33
      3.1.7 OpenHIP .......................................................................... 34
      3.1.8 Conclusion about Implementations .......................................... 37
   3.2 HIP vs. MIP Comparison ................................................................. 37
3.2.1 Performance ........................................... 38
3.2.2 Security ............................................. 38
3.2.3 Deployment .......................................... 39
3.2.4 Scalability .......................................... 40
3.2.5 Robustness .......................................... 40
3.2.6 Summary ............................................ 41
3.2.7 HIP vs. MIP Conclusions ......................... 41
3.2.8 Conclusions ......................................... 42
3.3 SIP Application ........................................ 42
3.3.1 Objectives .......................................... 42
3.3.2 Motivation ......................................... 42
3.3.3 Description of the Application ................. 42
3.3.4 Results ............................................ 49
3.3.5 Conclusions about the application ........... 52
3.4 Integrated operation of the testbed with the application ... 52
3.4.1 Results ............................................ 53

4 Conclusions ............................................... 59
4.1 Objectives Review ..................................... 59
4.2 Results ................................................ 59
4.3 Future Work ........................................... 60

A Host Identity Protocol Manual .................................. 61
A.1 Introduction ........................................... 61
A.2 Terminology ........................................... 62
A.2.1 Host Identity Namespace ................................... 62
A.2.2 Host Identifier - HI ................................... 62
A.2.3 Host Identity Tag- HIT ................................ 63
A.2.4 Local Scope Identifier- LSI .................................. 64
A.3 IPSec Background ....................................... 65
A.3.1 Diffie-Hellman ....................................... 65
A.3.2 IPSec Security Associations - SAs ................... 65
A.4 HIP Operation .......................................... 66
A.4.1 Base Exchange ....................................... 66
A.4.2 Simplified HIP State Diagram ....................... 71
A.4.3 The Rendesvouz Server Mechanism ............... 71
A.4.4 Description of the HIP packets ................... 73
A.4.5 Mobility ............................................ 76
A.4.6 Integration with DNS .................................. 78
A.4.7 Multihoming ......................................... 85
A.5 Conclusions ............................................ 87

B Log files .................................................. 88
B.1 HIP Basic Base Exchange ................................ 88
B.1.1 I1- HIP Initiator Packet ............................. 88
B.1.2 R1- HIP Responder Packet ............................ 90
B.1.3 I2- Second HIP Initiator Packet ................... 92
B.1.4 R2- Second HIP Responder Packet .................. 94
B.2 Mobility .................................................. 95
B.2.1 First HIP Update Packet ............................. 96
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.2.2</td>
<td>Second HIP Update Packet</td>
<td>98</td>
</tr>
<tr>
<td>B.2.3</td>
<td>Third HIP Update Packet</td>
<td>100</td>
</tr>
<tr>
<td>B.3</td>
<td>Base Exchange with RVS</td>
<td>101</td>
</tr>
<tr>
<td>B.3.1</td>
<td>I1- HIP Initiator Packet</td>
<td>101</td>
</tr>
<tr>
<td>B.3.2</td>
<td>R1- HIP Responder Packet</td>
<td>102</td>
</tr>
<tr>
<td>B.3.3</td>
<td>I2- Second HIP Initiator Packet</td>
<td>104</td>
</tr>
<tr>
<td>B.3.4</td>
<td>R2- Second HIP Responder Packet</td>
<td>106</td>
</tr>
<tr>
<td>B.3.5</td>
<td>SIP - INVITE Request</td>
<td>107</td>
</tr>
<tr>
<td>B.3.6</td>
<td>SIP - 180 (Ringing) Response</td>
<td>109</td>
</tr>
<tr>
<td>B.3.7</td>
<td>SIP - 200 (OK) Response</td>
<td>111</td>
</tr>
<tr>
<td>B.3.8</td>
<td>SIP - ACK Request</td>
<td>112</td>
</tr>
<tr>
<td>B.3.9</td>
<td>SIP - REFER Request</td>
<td>113</td>
</tr>
<tr>
<td>B.3.10</td>
<td>SIP - 202 (Accepted) Response</td>
<td>114</td>
</tr>
<tr>
<td>C</td>
<td>Configuration Files</td>
<td></td>
</tr>
<tr>
<td>C.1</td>
<td>HIP4BSD - hipconf.xml</td>
<td>115</td>
</tr>
<tr>
<td>C.2</td>
<td>OpenHIP - hip.conf</td>
<td>115</td>
</tr>
<tr>
<td>C.3</td>
<td>OpenHIP - my_host_identities</td>
<td>117</td>
</tr>
<tr>
<td>C.4</td>
<td>OpenHIP - known_host_identities</td>
<td>119</td>
</tr>
<tr>
<td>D</td>
<td>How-Tos</td>
<td></td>
</tr>
<tr>
<td>D.1</td>
<td>Installation of the JMF on Linux</td>
<td>120</td>
</tr>
<tr>
<td>D.2</td>
<td>Installation of the HIP4BSD on Linux</td>
<td>121</td>
</tr>
<tr>
<td>D.3</td>
<td>Installation of the HIPL on Linux</td>
<td>125</td>
</tr>
</tbody>
</table>
There is currently a semantic overload of the IP Address namespace as it acts as
locator - referring the attachment point of an interface to the network and as identifier,
naming the interfaces as a way of identification in the network topology. This is one
of the major barriers to the deployment of mobility and multi-homing in the current
Internet as it is hard to perform dynamic readdressing of the entities.

HIP proposes the decoupling of the network layer from the transport layer, adding
a new layer to the networking stack that maps the host identifiers to network addresses
and vice-versa. This achieves the main architectural goal of HIP: the separation of
identifiers from locators, which is the main driving force behind mobility and multi-
大众.

SIP is widely deployed and supported as a signaling protocol for the application
layer. By its architecture, it provides the ability for a user to be reachable at any termi-
nal, at any network attachment point using the same URI.

This document reports on the experiments developed with HIP which provided an
enriching learning experience. It describes the development of a SIP based application
with the proposed goal of experimenting with the mobility capabilities of SIP and fi-
nally it also describes the integration of that application with the testbed as a combined
mobility solution.
Chapter 1

Introduction

1.1 Evolution of mobile communications

Mobility is one of the most important concepts in communications nowadays. Since the development of mobile terminals, the quantity and quality of services provided to the user has been escalating at an unpredictable rate. Starting in the 80's with a first generation of analog cellphones, we are now moving towards a fourth generation technology which aims to provide ubiquitous communications to all users. Theoretically, in a near future, the user should be able to access any service from any terminal, anywhere, putting the focus on mobility and ubiquity. The current Internet architecture, however, is still limited in this aspect as it lacks the support for seamless mobility. In this work, we experiment possible solutions for session and terminal mobility and compare approaches to the mobility problem.

1.2 Objectives

With this project we aim to build a test-bed that would permit the study of a full mobility solution.

A full mobility solution should allow a user to communicate without breaks while moving, always choosing the network that offers the best service. It should also allow the user to change terminals seamlessly during a communication (i.e. Session Mobility). To accomplish network mobility we chose to experiment HIP, an emerging protocol based on the concept of indirection architectures[25]. These experiments require the implementation of a HIP test-bed.

To enable session mobility we propose a solution based on SIP. This will require the development of a SIP-based application which will implement SIP primitives designed for session mobility. Therefore, the objectives are:

- Implementation of a fully working HIP test-bed, experimenting with all of the available implementations of the protocol
- Development of a HIP manual for future reference
- Development of an application based on SIP that will allow for application mobility between terminals
1.3 Expected Results

The integrated implementation of the HIP test-bed with the SIP application should permit a high degree of experimentation with a full mobility solution. The HIP test-bed should permit a empirical learning experience of the protocol, contributing to the understanding of the strengths and weaknesses of HIP. The test-bed should comprise the following characteristics:

- RVS usage for mobility purposes that can be achieved either by the use of a dedicated HIP host or a distributed public solution like a DHT server.
- DNS integration for resolving SIP addresses

The SIP application should allow an understanding about the possibilities of SIP in an application mobility scenario. The application, built using a Java Platform, should provide a hands-on approach to the SIP protocol and its implementation procedures. It should also allow to evaluate the adequation of some of the SIP primitives (not initially conceived for application mobility) to our scenario. The SIP application should provide means for:

- the establishment of a SIP session.
- the transfer of the established session.
- the correct termination of the established session.

1.4 Document Structure

This document consists on 4 chapters:

1. An Introduction chapter which presents the subject of the work and where we clearly state the objectives and expected results of our work.

2. A State of the Art chapter. Here we’ll present some of the current mobility associated technologies that are pertinent to our work, namely:

   - MIP - Some of the aspects of MIP will be presented providing some terms for comparison with HIP.
   - HIP - In this section HIP will be explained superficially. This section is oriented to the reader who wants to understand the basic concepts and operation of HIP.
     For a deeper comprehension of HIP, an appendix (Appendix A) was elaborated. This appendix is self-contained and can be used for future reference.
   - SIP - Here we’ll refer the important components of SIP with particular emphasis on those more relevant to our work.

3. A Work Description chapter where we’ll explain thoroughly the work done, show the results and draw some conclusions.

4. A Conclusions chapter were a final analysis of the project will be done
Chapter 2

State of the Art

The Internet is built from three major components: computing platforms (end-points), packet transport (i.e., internetworking) infrastructure, and services (applications). The Internet exists to service two particular components: people and robotic services. All these components need to be named in order to interact in a scalable manner.

There are two principal namespaces in use in the Internet for these components: Domain Names and IP addresses. Domain Names provide hierarchically assigned names for some computing platforms and some services. Each hierarchy is delegated from the level above. There is no anonymity in Domain Names. Email, HTTP, and SIP addresses all reference Domain Names.

IP addresses have, currently, two different functions: they act as locators and identifiers of a network interface. As locators they refer to the interface’s attachment point to the network. As identifiers, they attribute a name to an interface on the network topology. Attributing both locator and end-point identifier functions to the IP addresses is one of the major barriers to the deployment of mobility and multi-homing in the current Internet as it is hard to perform dynamic readdressing of the entities.

Thus, there are three critical deficiencies with the current namespaces: first, dynamic readdressing cannot be directly managed. Second, anonymity is not provided in a consistent, trustable manner and finally, authentication for systems and datagrams is not provided. All of these deficiencies arise because computing platforms are not well named with the current namespaces.
2.1 Mobile IP - MIP

Mobile IP was originally developed as an extension to IPv4 protocols (MIPv4) [3]. In contrast, Mobile IPv6 has been developed as an integral part of IPv6 (MIPv6), which as of this writing has nearly been approved as a proposed standard [10]. It is the consensus of the IETF Mobile IP working group that MIPv6 is an improved version of MIPv4. Both MIPv4 and MIPv6 offer a mode of operation described below as "Mobile IP Through a Home Agent." Additionally, MIPv6 specifies a second mode of operation, "Mobile IP with Route Optimization." For state of the art purposes, we'll simply consider the possibilities offered by MIPv6. Thus, from now on, MIPv6 will be referred to as Mobile IP - MIP. This section will focus particularly on some MIP characteristics that will establish a base for comparison with HIP, presented in the Work Description chapter.

2.1.1 Introduction

Mobile IPv6 allows a mobile node to move from one link to another without changing the mobile node's "home address". Packets may be routed to the mobile node using this address regardless of the mobile node's current point of attachment to the Internet. The mobile node may also continue to communicate with other nodes (stationary or mobile) after moving to a new link. The movement of a mobile node away from its home link is thus transparent to transport and upper-layer protocols, like TCP, and applications.

2.1.2 Terminology

- Home Address - HA
  A unicast routable address assigned to a mobile node, used as the permanent address of the mobile node. This address is within the mobile node's home link. Standard IP routing mechanisms will deliver packets destined for a mobile node's home address to its home link. Mobile nodes can have multiple home addresses, for instance when there are multiple home prefixes on the home link.

- Mobile Node - MN
  A node that can change its point of attachment from one link to another, while still being reachable via its home address.

- Correspondent Node - CN
  A peer node with which a mobile node is communicating. The correspondent node may be either mobile or stationary.

- Care-of Address - CoA
  A unicast routable address associated with a mobile node while visiting a foreign link. The subnet prefix of this IP address is a foreign subnet prefix. Among the multiple care-of addresses that a mobile node may have at any given time (e.g., with different subnet prefixes), the one registered with the mobile node's home agent for a given home address is called its "primary" care-of address.

- Home Agent - HA
  A router on a mobile node's home link with which the mobile node has registered its current care-of address. While the mobile node is away from home, the home agent intercepts packets on the home link destined to the mobile node's home...
address, encapsulates them, and tunnels them to the mobile node’s registered care-of address.

- Binding
  The association of the home address of a mobile node with a care-of address for that mobile node, along with the remaining lifetime of that association.

2.1.3 Basic Operation

Mobile IP through a Home Agent

Mobile IP through a Home Agent (MIP-HA) is based on the paradigm of indirection architecture [25]. The Home Agent acts as the indirection point allowing a mobile node to be always accessible through the relaying of traffic from the Correspondent Node to the Mobile Node. A basic mobility scenario in this case would be:

- The MN moves to a new network, where it self-configures a new address - Care-of Address
- The MN registers the CoA with its Home Agent using a BindingUpdate message
- The HA registers the new MN’s address and confirms the registration with a BindingAcknowledgement message

When a Correspondent Node sends a packet to the Mobile Node:

- The HA intercepts the packet and sends it to the MN’s CoA through a tunnel

When the MN wants to communicate with the CN:

- The MN sends the packet through a tunnel to the HA
- The HA relays the packet to the CN

Mobile IP with Route Optimization

Route Optimization extensions were introduced to improve scalability and reliability and reduce the network load introduced by the MIP-HA mobility process.

These extensions assume default use of home agent-based operation, but also allow a mobile node to notify a correspondent node directly of the mobile nodes current address, to permit the correspondent node to deliver packets directly to the mobile node. For a Correspondent Node to communicate with the Mobile Node:

- The CN consults its Binding Cache. The Bindings Cache contains the current CoA of the MN. When the MN changes address, it uses a BindingUpdate message to inform the CN that updates the Binding Cache.
- If there's no entry on the Bindings Cache, the packet is sent normally. But if there’s an entry the packet is sent directly to the CoA. In the latter case, the CN includes in the packet a Routing Header to carry the address of the MN’s HA.

When the MN wants to communicate with the CN:
• The MN uses its CoA as the source address
• The MN includes a Destination Header with information about its Home Agent
• When receiving a packet, the CN substitutes the CoA by the HA address as the source address. This is necessary so mobility procedures are transparent to upper layer protocols.

2.1.4 Security

Return Routability - RR

Return Routability is the name of the basic mechanism deployed by Mobile IPv6 route optimization security design. RR is based on the idea that a node should be able to verify that there is a node that is able to respond to packets sent to a given address. The check yields false positives if the routing infrastructure is compromised or if there is an attacker between the verifier and the address to be verified. With these exceptions, it is assumed that a successful reply indicates that there is indeed a node at the given address, and that the node is willing to reply to the probes sent to it. The RR’s packet flow is depicted in fig. 2.1.

![Figure 2.1: Return Routability Packet Flow](image)

- First, the mobile node sends two packets to the Correspondent Node: a Home Test Init (HoTI) packet is sent through the Home Agent, and a Care-of Test Init (CoTI) directly.
- The Correspondent Node replies to both of these independently by sending a Home Test (HoT) in response to the Home Test Init and a Care-of Test (CoT) in response to the Care-of Test Init.
- Finally, once the Mobile Node has received both the Home Test and Care-of Test packets, it sends a Binding Update to the Correspondent Node.

**IPSec**

As seen in section 2.1.3 Mobile IPv6 tunnels payload packets between the mobile node and the home agent in both directions. This tunneling uses IPv6 encapsulation [4].
Where these tunnels need to be secured, they are replaced by IPsec tunnels. Mobile IPv6 base specification uses IPsec Encapsulating Security Payload (ESP) [14] to protect control traffic between the home agent and the mobile node.

The MIPv6 specification defines the interaction between outbound Mobile IPv6 processing and outbound IPsec processing for packets sent by a mobile node while away from home. This procedure is described in the following steps:

- The packet is created by higher layer protocols and applications (e.g., by TCP) as if the mobile node were at home and Mobile IPv6 were not being used.

- Determine the outgoing interface for the packet. (Note that the selection between reverse tunneling and route optimization may imply different interfaces, particularly if tunnels are considered interfaces as well.)

- As part of outbound packet processing in IP, the packet is compared against the IPsec security policy database to determine what processing is required for the packet [15].

- If IPsec processing is required, the packet is either mapped to an existing Security Association (or SA bundle), or a new SA (or SA bundle) is created for the packet, according to the procedures defined for IPsec.

- Since the mobile node is away from home, it will be using either reverse tunneling or route optimization to reach the correspondent node.

  If reverse tunneling is used, the packet is constructed in the normal manner and then tunneled through the home agent.

  If route optimization is in use, the mobile node inserts a Home Address destination option into the packet, replacing the Source Address in the packet’s IP header with the care-of address used with this correspondent node, as described in section A.3. The Destination Options header in which the Home Address destination option is inserted must appear in the packet after the routing header, if present, and before the IPsec (AH [13] or ESP [14]) header, so that the Home Address destination option is processed by the destination node before the IPsec header is processed.

- Finally, once the packet is fully assembled, the necessary IPsec authentication (and encryption, if required) processing is performed on the packet, initializing the Authentication Data in the IPsec header.

- RFC 2402 treatment of destination options is extended as follows. The AH authentication data must be calculated as if the following were true:

  The IPv6 source address in the IPv6 header contains the mobile node’s home address,

  The Home Address field of the Home Address destination option contains the new care-of address.

- This allows, but does not requires, the receiver of the packet containing a Home Address destination option to exchange the two fields of the incoming packet to reach the above situation, simplifying processing for all subsequent packet headers. However, such an exchange is not required, as long as the result of the authentication calculation remains the same.
2.2 Host Identity Protocol - HIP

2.2.1 Introduction

The current Internet uses two global namespaces: domain names and IP addresses. The first namespace - domain names - has a single use acting as symbolic identifiers for sets of numeric IP addresses. IP addresses form the Internet's second global namespace. They have two uses: First, they are topological locators for network attachment points, addressing a specific location in the network topology. Their second use is as identifiers for the network interfaces - and thus nodes - naming those interfaces on the network topology.

The Host Identity Protocol (HIP) is a multi-addressing and mobility solution for the IPv4 and IPv6 Internet with always-on security policy. HIP adds a new Host Identity (HI) layer between the transport and network layers [18]. The new layer maps the host identifiers to network addresses and vice versa. This achieves the main architectural goal of HIP: the separation of identifiers from locators, which is the main driving force behind mobility and multihoming. The new layer also provides host-to-host authentication, identifying the hosts using their public keys or the hash values of public keys (HIDs). The new architecture will provide intrinsic security and support for IP-layer mobility and multi-homing [17].

HIP provides tools for implementing authentication and encryption, mobility and multi-homing, and protection from Denial-of-Service.

2.2.2 Terminology

Host Identity Namespace

A name in the Host Identity Namespace - a Host Identifier (HI) - represents a statistically globally unique name for naming any system with an IP stack. This identity is normally associated with, but not limited to, an IP stack. A system can have multiple identities, some 'well known', some unpublished or 'anonymous'. A system may self-assert its own identity, or may use a third-party authenticator like DNS Security (DNSSEC) [1], Pretty Good Privacy (PGP) [5], or X.509 [9] to 'notarize' the identity assertion. It is expected that the Host Identifiers will initially be authenticated with DNSSEC and that all implementations will support DNSSEC as a minimal baseline.

In theory, any name that can claim to be 'statistically globally unique' may serve as a Host Identifier. However a public key of a 'public key pair' was chosen for HIP. A public-key-based HI can authenticate the HIP packets and protect them from man-in-the-middle attacks. Since authenticated datagrams are mandatory to provide much of HIP's DoS protection, the Diffie-Hellman exchange in HIP has to be authenticated. Thus, only public key HI and authenticated HIP messages are supported in practice.

2.2.3 Host Identifier - HI

The Host Identifier is the public part of a public/private key pair that univocally identifies a host in the Host Identity Namespace.

The Host Identity adds two main features to Internet protocols: The first is a decoupling of the internetworking and transport layers. This decoupling will allow for independent evolution of the two layers. In addition, it can provide end-to-end services over multiple internetworking realms. The second feature is host authentication. Because the Host Identifier is a public key, this key can be used for authentication in
security protocols like IPsec. The possession of the private part of the public/private key pair defines the Identity itself. If the private key is possessed by more than one node, the Identity can be considered to be a distributed one.

The actual Host Identities are never directly used in any Internet protocols. Although it can be kept in DNS records, and passed during communication, for usability purposes, two representations of the HI were created: the Host Identity Tag (HIT) and the Local Scope Identifier (LSI).

**Host Identity Tag- HIT**

A Host Identity Tag is a 128-bit representation for a Host Identity.

Its format, similar to IPv6 (e.g. 1100:/:/8) but without any structure, is created by taking a cryptographic hash over the corresponding Host Identifier. There are two advantages of using a hash over using the Host Identifier in protocols. First, its fixed length makes for easier protocol coding and also better manages the packet size cost of this technology. Second, it presents the identity in a consistent format to the protocol independent of the cryptographic algorithms used.

In the HIP packets, the HITs identify the sender and recipient of a packet. Consequently, a HIT should be unique in the whole IP universe as long as it is being used. In the extremely rare case of a single HIT mapping to more than one Host Identity, the Host Identifiers (public keys) will make the final difference. If there is more than one public key for a given node, the HIT acts as a hint for the correct public key to use.

**Local Scope Identifier- LSI**

A Local Scope Identifier is a 32-bit localized representation for a Host Identity.

In many environments, 128 bits is still considered large. For example, currently used IPv4 based applications are constrained with 32-bit address fields. Another problem is that the cohabitation of IPv6 and HIP might require some applications to differentiate an IPv6 address from a HIT. So, the purpose of an LSI is to facilitate using Host Identities in existing protocols and APIs. There are two types of such LSIs: 32-bit IPv4-compatible ones and 128-bit IPv6-compatible ones. The LSI provides a compression of the HIT with only a local scope so that it can be carried efficiently in any application level packet and used in API calls. The LSIs do not have the same properties as HITs (i.e., they are not self-certifying nor are they as unlikely to collide - hence their local scope), and consequently they must be used more carefully.

By default, the LSI is obtained using concatenating the binary value of 0000 0001 with the 24 LSB of the HIT. This produces a LSI with the format 1.xxx.xxx.xxx. The biggest advantages of LSI over HIT are its size and ability to work with legacy APIs. Its major disadvantage is its localized application.

### 2.2.4 Basic Operation

**Base Exchange**

The HIP base exchange serves to manage the establishment of state between an Initiator and a Responder.

By definition, the system initiating a HIP exchange is the Initiator, and the peer is the Responder. This distinction is forgotten once the base exchange completes, and either party can become the Initiator in future communications. The Base Exchange is used to establish a pair of IPsec security associations (SA) (see section A.3.2 for IPSec
details) between two HIP systems, also called HIP hosts. After the the Base Exchange is finished, the data is encapsulated using, by default, IPSec’s ESP. Thus, we have two different network stacks: one for the signaling plane with HIP between layers 2 and 3 and another for the data plane with an ESP layer on the same position.

Figure 2.2: On the left: Networking Stack - Control Plane. On the right: Networking Stack - Data Plane (Using ESP)

The first packet, I1, initiates the exchange, and the last three packets, R1, I2, and R2, constitute a standard authenticated Diffie-Hellman (see section A.3.1) key exchange for session key generation. During the Diffie-Hellman key exchange, a piece of keying material is generated. The HIP association keys are drawn from this keying material. If other cryptographic keys are needed, e.g., to be used with ESP, they are expected to be drawn from the same keying material.

Figure 2.3: Base Exchange

Rendezvous Server Mechanism

HIP introduces a new mechanism based on the paradigm of the indirection architectures [25].

This paradigm assumes the existence of a physical or a logical indirection point interposed between the sender and the receiver(s) that relays the traffic between them. By communicating through the indirection point rather than directly to the end-host, a sender can abstract away the location and the number of receivers.
The Rendezvous Server Operation: A HIP node may want to be reachable to future correspondent peers that are unaware of its location change. The HIP architecture introduces Rendezvous Servers (RVS) with whom a HIP node may register its host identity tags (HITs) and current IP addresses. An RVS relays HIP packets arriving for these HITs to the node’s registered IP addresses.

![Figure 2.4: Base Exchange with RVS](image)

Mobility

Mobility in HIP can be split into two contexts:

- **Session Mobility**, that addresses the mobility scenario during a communication session.
- **Global Mobility**, which refers to pre-session mobility. In this scenario, mobility support is provided by the use of a RVS, as seen on section A.4.3.

**Session Mobility:** When a host moves to another address, it notifies its peer of the new address by sending a HIP UPDATE packet containing a LOCATOR parameter. This UPDATE packet is acknowledged by the peer, and is protected by retransmission. The peer can authenticate the contents of the UPDATE packet based on the signature and keyed hash of the packet.

When using ESP Transport Format [14], the host may at the same time decide to rekey its security association and possibly generate a new Diffie-Hellman key (see section A.3 for IPsec details). All of these actions are triggered by including additional parameters in the UPDATE packet, as defined in the base protocol specification [21] and ESP extension [11].

**Global Mobility:** Global mobility, defined as pre-communication mobility, is implemented in HIP by using the RVS mechanism discussed in section 2.2.4.

The mobile host must always keep his RVS up to date on his current location. This way, a host wanting to communicate with the mobile host can always send the I1 packet directly to the RVS, abstracting away from the mobile host’s locator.
Integration with DNS

HIP introduces a new HIP DNS Resource Record to store Rendezvous Server (RVS), Host Identity (HI) and Host Identity Tag (HIT) information. This allows for an Initiator to query a DNS server as a way to obtain the Responder’s RVS address. This procedure is considered the be the most common way to start a communication in a full deployment situation.

An initiator willing to associate with a node would typically issue the following queries:

\[
\text{QNAME=www.example.com, QTYPE=HIP}
\]

Which returns a DNS packet with RCODE=0 and one or more HIP RRs with the HIT and HI (e.g., HIT-R and HI-R) of the responder in the answer section, but no RVS.

\[
\text{QNAME=www.example.com, QTYPE=A}
\]

Which returns a DNS packet with RCODE=0 and one or more A or AAAA RRs containing IP address(es) of the responder (e.g., IP-R) in the answer section. The initiator would then send an I1 to the responder’s IP addresses (IP-R), starting the Base Exchange.

Figure 2.6: Example of DNS integration with HIP
Multihoming

A (mobile or stationary) host may sometimes have more than one interface or global address. The host may notify the peer host of the additional interface or address by using the LOCATOR parameter. To avoid problems with the ESP anti-replay window, a host should use a different SA for each interface or address used to receive packets from the peer host.

When more than one locator - defined as a point of attachment to a network - is provided to the peer host, the host should indicate which locator is preferred. By default, the addresses used in the base exchange are preferred until indicated otherwise.

Although the protocol may allow for configurations in which there is an asymmetric number of SAs between the hosts (e.g., one host has two interfaces and two inbound SAs, while the peer has one interface and one inbound SA), it is recommended that inbound and outbound SAs be created pairwise between hosts. When an ESP_INFO arrives to rekey a particular outbound SA, the corresponding inbound SA should also be rekeyed at that time. When a host wants to communicate to its peer the presence of another locator ready for communication, the follow procedure is depicted in fig. 2.7 should be followed:

![Figure 2.7: Basic Multihoming scenario](image)

2.2.5 Conclusions

HIP presents itself as a promising technology. It is based on the paradigm of indirection architecture which has been a common approach for the current efforts to separate the identifiers for the locator.

HIP decouples the internetworking layer from the transport layer, allowing each to evolve separately. The decoupling makes end-host mobility and multi-homing easier, across IPv4 and IPv6 networks. HIs make network renumbering easier, and they also make process migration and clustered servers easier to implement. Furthermore, being cryptographic in nature, they provide the basis for solving the security problems related to end-host mobility and multi-homing. As disadvantages, HIP has many functional requirements, particularly alterations to existing infrastructures and the introduction of new ones.

HIP is still not as a complete solution as MIP, but this could be simply because of MIP's head start. HIP stands on good premises and seams to have potential to evolve.
As an objective of the project, a HIP manual was elaborated (see Appendix A). This manual provides an in-depth view of HIP, presenting all of the subjects referred in this State of the Art section dedicated to HIP in a more detailed fashion. However, it must be kept in mind that HIP and its implementations are still in a development phase, and thus, may present alterations to the descriptions in the manual in the near future.

### 2.3 Session Initiation Protocol - SIP

In this section will provide a basic overview of SIP, referring its most important aspects but focusing on the mobility characteristics that had high relevance in our work.

#### 2.3.1 Introduction

The Session Initiation Protocol [22] is an application-layer control protocol that can establish, modify, and terminate multimedia sessions such as Internet telephony calls. SIP can also invite participants to already existing sessions, such as multicast conferences. Media can be added to (and removed from) an existing session. SIP transparently supports name mapping and redirection services, which allows for personal mobility - users can maintain a single externally visible identifier regardless of their network location.

SIP supports five facets of establishing and terminating multimedia communications:

- **User location**: determination of the end system to be used for communication
- **User availability**: determination of the willingness of the called party to engage in communications
- **User capabilities**: determination of the media and media parameters to be used
- **Session setup**: "ringing", establishment of session parameters at both called and calling party
- **Session management**: including transfer and termination of sessions, modifying session parameters, and invoking service

#### 2.3.2 Terminology

- **User Agent Client - UAC**
  Caller application that initiates and sends SIP requests.
- **User Agent Server - UAS**
  Receives and responds to SIP requests on behalf of clients; accepts, redirects or refuses calls.
- **SIP Terminal**
  Supports real-time, 2-way communication with another SIP entity. Supports both signalling and media, similar to H.323 terminal. Is constituted by a UAC and possibly a UAS being, in this case, simply referred as a User Agent - UA
- **Proxy**
  Contacts one or more clients or next-hop servers and passes the call requests further. Contains UAC and UAS.
• Redirect Server
  Accepts SIP requests, maps the address into zero or more new addresses and
  returns those addresses to the client. Does not initiate SIP requests or accept
  calls.

• Location Server
  Provides information about a caller's possible locations to redirect and proxy
  servers. May be co-located with a SIP server.

• Digits in capabilities
  Refers to the functions that the user wants to invoke. For example, the client
  software might support videoconferencing but the user may only want to use
  audio conferencing. The user can always add functions during the session by
  issuing another INVITE request to other users on the link.

2.3.3 SIP and SIPS URIs

The "sip" and "sips:" schemes follow the guidelines in RFC 2396. They use a form
similar to the mailto URL, allowing the specification of SIP request-header fields
and the SIP message-body. This makes it possible to specify the subject, media type, or
emergency session initiated by using a URI on a web page or in an email message. Its
general form, in the case of a SIP URI, is:

    sip: user:password@host:port;uri-parameters?headers

The syntax for a SIPS URI is the same, except that the scheme is "sips" instead of
sip.

For further information on the SIP and SIPS URI parameters, refer to RFC3261,
section 1.1.

2.3.4 SIP’s REFER Extension

The SIP REFER extension[24] provides a mechanism allowing a party sending the
REFER to be notified of the outcome of the referenced request. This can be used to
enable any applications, including call transfer. This can be used to enable many
applications, including Call Transfer. For instance, if Alice is in a call with Bob, and
decides Bob needs to talk to Carol, Alice can instruct her SIP UA to send a SIP REFER
request to Bob's UA providing Carol's SIP Contact information. Assuming Bob has
given it permission, Bob's UA will attempt to call Carol using that contact. Bob's UA
will then report whether it succeeded in reaching the contact to Alice's UA.

To accomplish our purposes, we suggested minor changes in the use of this extension.
A party wanting to redirect a media stream or a call would send 2 REFER mes-
ges. The first message would go to the media Server informing it of a new address to
which the stream should be now transmitted. This information would be carried in the
Refer-To header. This message would have the same parameters previously negotiated
and would, this way, be recognized by the Server. A second REFER message would be
sent to the referred-to Client. This message would carry information about the Server's
address in the Refer-To header and a SDP message body with information related with
the media stream of call. This information would typically comprise the file that was
being streamed, for example and the type of media. This information would then be
used in an INVITE sent from the referred Client to the Server. The signaling procedure
for the case of a media stream is depicted in fig.2.8.
2.3.5 Operation

Signaling

SIP defines as Methods or Requests the signaling messages sent from a User Agent Client to a User Agent Server and as Responses the messages from the UAS to the UAC.

SIP Requests

<table>
<thead>
<tr>
<th>SIP Requests</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVITE</td>
<td>Invites a user to a call</td>
</tr>
<tr>
<td>ACK</td>
<td>Used to facilitate reliable message exchange for INVITEs</td>
</tr>
<tr>
<td>BYE</td>
<td>Terminates a connection between users or declines a call</td>
</tr>
<tr>
<td>CANCEL</td>
<td>Terminates a request, or search, for a user</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>Solicits information about a server’s capabilities</td>
</tr>
<tr>
<td>REGISTER</td>
<td>Registers a user’s current location</td>
</tr>
<tr>
<td>INFO</td>
<td>Used for mid-session signalling</td>
</tr>
</tbody>
</table>

SIP Responses
<table>
<thead>
<tr>
<th>Response</th>
<th>Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1xx</td>
<td>Informational</td>
<td>100 Trying, 180 Ringing</td>
</tr>
<tr>
<td>2xx</td>
<td>Successful</td>
<td>200 OK, 202 Accepted</td>
</tr>
<tr>
<td>3xx</td>
<td>Redirection</td>
<td>302 Moved Temporarily</td>
</tr>
<tr>
<td>4xx</td>
<td>Request Failure</td>
<td>404 Not Found, 482 Loop Detected</td>
</tr>
<tr>
<td>5xx</td>
<td>Server Failure</td>
<td>501 Not Implemented</td>
</tr>
<tr>
<td>6xx</td>
<td>Global Failure</td>
<td>603 Decline</td>
</tr>
</tbody>
</table>

**Operation Example**

As an overview of the operation, we'll present a typical example that shows the basic functions of SIP:

- Location of an end point
- Signal of a desire to communicate
- Negotiation of session
- Parameters to establish the session
- Teardown of the session once established

Figure 2.9 shows a typical example of a SIP message exchange between two users, Alice and Bob. Also shown are two SIP proxy servers that act on behalf of Alice and Bob to facilitate the session establishment. This typical arrangement is often referred to as the "SIP trapezoid" as shown by the geometric shape of the dotted lines in the figure.

SIP is based on an HTTP-like request/response transaction model. Each transaction consists of a request that invokes a particular method, or function, on the server and at least one response. In this example, the transaction begins with Alice's softphone sending an INVITE request addressed to Bob's SIP URI. INVITE is an example of a SIP request that specifies the action that the requestor (Alice) wants the server (Bob) to take. The INVITE request contains a number of header fields. Header fields are named attributes that provide additional information about a message. The ones present in an INVITE include a unique identifier for the call, the destination address, Alice's address, and information about the type of session that Alice wishes to establish with Bob. An example INVITE message is shown next:

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76a66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142
```
Alice uses a SIP application on her PC (referred to as a softphone) to call Bob on his SIP phone over the Internet. Alice "calls" Bob using his SIP identity, a type of Uniform Resource Identifier (URI) called a SIP URI. It has a similar form to an email address, typically containing a username and a host name. In this case, it is sip:bob@biloxi.com, where biloxi.com is the domain of Bob's SIP service provider. Alice has a SIP URI of sip:alice@atlanta.com. Alice might have typed in Bob's URI or perhaps clicked on a hyperlink or an entry in an address book. SIP also provides a secure URI, called a SIPS URI (see section 2.3.3 for the definition of SIP and SIPS URIs). An example would be sips:bob@biloxi.com. A call made to a SIPS URI guarantees that secure, encrypted transport (namely TLS) is used to carry all SIP messages from the caller to the domain of the callee. From there, the request is sent securely to the callee, but with security mechanisms that depend on the policy of the domain of the callee.

- Alice initiates the call. The softphone does not know the location of Bob or the SIP server in the biloxi.com domain. So, the softphone sends the INVITE (F1) to the SIP server that serves Alice's domain, atlanta.com. The address of the atlanta.com SIP server could have been configured in Alice's softphone, or it could have been discovered by DHCP, for example.

- The atlanta.com SIP server receives the INVITE request for future forwarding.

In this case, atlanta.com is a proxy server. The atlanta.com proxy server locates the proxyserver at biloxi.com, possibly by performing a particular type of DNS
(Domain Name Service) lookup to find the SIP server that serves the biloxi.com domain. As a result, it obtains the IP address of the biloxi.com proxy server and forwards, or proxies, the INVITE (F2) request there. Before forwarding the request, the atlanta.com proxy server adds an additional Via header field value that contains its own address (the INVITE already contains Alice's address in the first Via).

- The Proxy Server sends a 100 (Trying) response (F3) back to Alice's softphone. The 100 (Trying) response indicates that the INVITE has been received and that the proxy is working on her behalf to route the INVITE to the destination.

- The biloxi.com proxy server receives the INVITE (F2) and responds with a 100 (Trying) response (F5) back to the atlanta.com proxy server to indicate that it has received the INVITE and is processing the request. The proxy server consults a database, generically called a location service, that contains the current IP address of Bob.

- The biloxi.com proxy server adds another Via header field value with its own address to the INVITE and proxies it to Bob's SIP phone (F4).

- Bob's SIP phone receives the INVITE and alerts Bob to the incoming call from Alice so that Bob can decide whether to answer the call. Bob's SIP phone indicates this in a 180 (Ringing) response (F6), which is routed back through the two proxies in the reverse direction (F7, F8). Each proxy uses the Via header field to determine where to send the response and removes its own address from the top. As a result, although DNS and location service lookups were required to route the initial INVITE, the 180 (Ringing) response can be returned to the caller without lookups or without state being maintained in the proxies. This also has the desirable property that each proxy that sees the INVITE will also see all responses to the INVITE.

- When Alice's softphone receives the 180 (Ringing) response (F8), it passes this information to Alice, perhaps using an audio ringback tone or by displaying a message on Alice's screen.

- When Bob picks up the handset, his SIP phone sends a 200 (OK) response (F9, F10, F11) to indicate that the call has been answered. The 200 (OK) contains a message body with the SDP media description of the type of session that Bob is willing to establish with Alice. As a result, there is a two-phase exchange of SDP messages: Alice sent one to Bob, and Bob sent one back to Alice. This two-phase exchange provides basic negotiation capabilities and is based on a simple offer/answer model of SDP exchange. If Bob did not wish to answer the call or was busy on another call, an error response would have been sent instead of the 200 (OK), which would have resulted in no media session being established.

- Finally, Alice's softphone sends an acknowledgement message, ACK (F12), to Bob's SIP phone to confirm the reception of the final response (200 (OK)). In this example, the ACK is sent directly from Alice's softphone to Bob's SIP phone, bypassing the two proxies. This occurs because the endpoints have learned each other's address from the Contact header fields through the INVITE/200 (OK) exchange, which was not known when the initial INVITE was sent. The lookups performed by the two proxies are no longer needed, so the proxies drop out of
the call flow. This completes the INVITE/200/ACK three-way handshake used to establish SIP sessions.

- Alice and Bob's media session has now begun, and they send media packets using the format to which they agreed in the exchange of SDP. In general, the end-to-end media packets take a different path from the SIP signaling messages.

- At the end of the call, Alice disconnects (hangs up) first and generates a BYE (F13) message. This BYE is routed directly to Bob's softphone, again bypassing the proxies. Bob confirms receipt of the BYE with a 200 (OK) response (F14), which terminates the session and the BYE transaction. No ACK is sent - an ACK is only sent in response to a response to an INVITE request.

Please refer to section 2.3.5 for a complete list and description of SIP Requests and Responses.

The details of the session, such as the type of media, codec, or sampling rate, are not described using SIP. Rather, the body of a SIP message contains a description of the session, encoded in some other protocol format. One such format, topically used, is the Session Description Protocol (SDP) (RFC 2327).

2.3.6 Mobility

The mobility concept associated with SIP can be looked upon from different perspectives.

- SIP, by its architecture supports Personal Mobility, *i.e.*, a user can be found independently of location or network device.

- SIP also has mechanisms that allow Network mobility, *i.e.*, the ability for a device to move between different points of attachment to the network.

- Finally, SIP has been developing mechanisms that permit transferring an ongoing session between terminals, which we refer in this section as Application Mobility.

**Personal Mobility**

An user can register with a Location Server permitting that, through the use of Proxy Server or Redirect Servers, a call can always be addressed to that user using always the same SIP URI independently of the terminal at which the user can be currently reached. This procedure is known as SIP transaction in redirection mode and is depicted in Figure 2.10.

**Network Mobility**

Network Mobility can be subdivided (as in HIP or MIP) into pre-communication mobility, referred as Global Mobility and as post-establishment of session mobility, called Session Mobility.
Figure 2.10: SIP transaction in redirection mode

- Global Mobility
  In order to support IP mobility, i.e., the change of the underlying IP Address, it is necessary to add to SIP the ability for a terminal to move between network attachment points. It is assumed that the mobile host belongs to a home network, on which there is a SIP server (e.g., a SIP redirect server), which receives registrations from the mobile host each time it changes location. This is similar to Home Agent registration in Mobile IP. When the Correspondent Node (as defined in MIP, section 2.1.2) acting as a UAC sends an INVITE to the mobile host, the redirect server checks the current information about the mobile host’s location and redirects the INVITE there. Global mobility is depicted in Figure 2.11

Figure 2.11: Global Mobility

- Session Mobility

24
If the mobile host moves during a session, it must send a new INVITE to the correspondent host using the same call identifier as in the original call setup. It should put the new IP address in the Contact field of the SIP message, which tells the correspondent host where it wants to receive future SIP messages. To redirect the data traffic flow, it indicates the new address in the SDP field, where it specifies transport address. Session mobility is shown on Fig. 2.12

Figure 2.12: Session Mobility

Application Mobility

Application Mobility explores the possibility of transferring the ongoing session between two terminals. This scenario would use the REFER extension to SIP [24]. As this scenario is a substantial part of our work, it will be explained in the Work Description chapter.

2.4 State of the Art Conclusions

Mobile IP presents itself as a very complete technique in a network mobility scenario. Although its first installment - MIP through a Home Agent - proposes a solution that would pose problems in terms of scalability and reliability as well as network load, MIP’s evolution through Route Optimization extensions provides a procedure that should allow some evolution without the constrains previously referred. The continuous integration of extensions to the protocol has permitted MIP to keep responding to emerging necessities like security.

At the application level, SIP is an important protocol that is becoming widely deployed. It provides key signaling elements, which can turn a voice over IP network into a true IP communications network - a network capable of delivering next generation converged services. SIP could, on its own, provide personal, network and application mobility, although, its application will probably be focused in personal and application mobility, leaving network mobility problems to be addressed by lower-level protocols.
HIP decouples the internetworking layer from the transport layer, allowing each to evolve separately. The decoupling makes end-host mobility and multi-homing easier, also across IPv4 and IPv6 networks. HIs make network renumbering easier, and they also make process migration and clustered servers easier to implement. Furthermore, being cryptographic in nature, they provide the basis for solving the security problems related to end-host mobility and multi-homing. As disadvantages, HIP has many functional requirements, particularly alterations to existing infrastructures and the introduction of new ones.

HIP is still not a complete solution as MIP, but this could be simply because of MIP's head start. HIP stands on good premises and seems to have a lot of potential to evolve.
Chapter 3

Work Description

3.1 HIP Test bed

3.1.1 Objectives

The HIP testbed had as main objective to permit an empirical study of HIP (Appendix A). The experiments with the test-bed should comprise all of the available implementations of the protocol. The experiences should permit to comprehend the generic operation of all implementations and consequently, the protocol. Also, because HIP is still in an early development phase, it would be important to understand the different approaches of each implementation to the protocol, each one with their advantages and disadvantages. These tests should allow to choose one implementation that, by the end of our project, could be integrated with the second part of our work - SIP mobility. Finally, the study of HIP should result on the elaboration of a HIP manual for future reference.

3.1.2 Motivation

HIP as been regarded as a promising protocol that would provide a new solution for network mobility. Its particular characteristics (discussed in Appendix A) like the introduction of a new layer in the network stack and its architecture, based on the paradigm on indirection architecture, made unavoidable the need to understand HIP’s advantages and application possibilities.

3.1.3 Basic Test-bed

The following description is common to all of the implementations presented ahead. The visible operation of HIP is similar to all of them. The main differences between them are in how the implementation is installed in the machine and in the dependencies that must be assured. The basic HIP test-bed must comprise a minimum of 2 HIP Hosts. These basic test-bed experiments had the simple purpose of making us understand the basic operation of HIP, namely, the Base Exchange.
The Base Exchange

The Base Exchange, as explained in section A.4.1 is a signaling procedure that be used to establish state between two hosts. A log capture of this procedure would have the appearance of fig. 3.1.

Figure 3.1: Base Exchange - Ethereal log

The detailed logs can be consulted in appendix B.1.

Mobility

The session mobility procedure, as explained on section A.4.5, is constituted by 3 UPDATE messages that inform the Peer host of the mobile host’s new location. In this case, the Mobile Host requests the SA rekeying. This particular procedure is explained in section A.4.5 under the paragraph "Mobility with single SA pair (peer-initiated rekey)“. A log capture of this procedure is shown on fig. 3.2.

In the log we can see the Base Exchange being established and then the 3 UPDATE messages of the mobility procedure. Because the Initiator requests the SA rekeying, he sends the value of the "Old SPI" and "New SPI". The "Old SPI" is the numeric identifier of the previous incoming Security Association and the "New SPI" is the numerical identifier of the new incoming Security Association (see detailed information about SPIs in section A.3.2).

The detailed logs can be consulted in appendix B.2.
<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>269</td>
<td>2006-03-09 12:10:04.646082</td>
<td>172.16.0.147</td>
<td>172.16.0.102</td>
<td>HIP II (HIP Initiator Packet)</td>
</tr>
<tr>
<td>270</td>
<td>2006-03-09 12:10:04.646140</td>
<td>172.16.0.102</td>
<td>172.16.0.147</td>
<td>HIP II (HIP Responder Packet)</td>
</tr>
<tr>
<td>314</td>
<td>2006-03-09 12:10:04.767840</td>
<td>172.16.0.147</td>
<td>172.16.0.102</td>
<td>HIP II (Second HIP Initiator Packet)</td>
</tr>
<tr>
<td>315</td>
<td>2006-03-09 12:10:04.767977</td>
<td>172.16.0.102</td>
<td>172.16.0.147</td>
<td>HIP II (Second HIP Responder Packet)</td>
</tr>
<tr>
<td>367</td>
<td>2006-03-09 12:10:04.983270</td>
<td>172.16.0.147</td>
<td>172.16.0.102</td>
<td>HIP UPDATE (HIP Update Packet)</td>
</tr>
<tr>
<td>372</td>
<td>2006-03-09 12:10:38.338363</td>
<td>172.16.0.102</td>
<td>172.16.0.147</td>
<td>HIP UPDATE (HIP Update Packet)</td>
</tr>
</tbody>
</table>

Frame 371 (200 bytes on wire, 200 bytes captured)
Ethernet II, Src: 00:50:56:00:00:00 (00:00:00:00:00:00), Dst: 00:50:56:00:00:00 (00:00:00:00:00:00)
HIP Identify Protocol
Payload: 39 bytes
Version: 5, Reserved: 0
Checksum: 0x2601 (correct)
Source's HIT: 0x50560000 (correct)
Receiver's HIT: 0x50560000 (correct)
HIP Parameters
- LISP 256 (type=0x05, length=256)
  - Reserved: 0x0000
  - Keymap Index: 0x0900
  - Keyset: 0x5f53f6de
  - New Keyset: 0x34a3b077
- SEQ (type=305, length=4)
- HMAC (type=61305, length=90)
- HIP_KEY (type=01107, length=120)

Figure 3.2: Mobility - Ethereal log

3.1.4 Test-bed with RVS

The HIP with RVS test-bed requires a minimum of 3 hosts - An Initiator, a Responder and a RVS.

The Base Exchange

In the base exchange procedure with a RVS, a packet is sent to the the RVS's IP but with the Responder's HIT. As described in section A.4.3, the packet in relayed from the RVS to the Responder. A log captured in this situation is shown on fig.3.3

As it can be seen from the log messages, the Initiator with the IP 172.16.0.147 sends the first base exchange packet - II - to the host with IP 172.16.0.102. This host plays the role of RVS and will relay the packet to the Responder after verifying that the packet's HIT is not one of his own.

The next message - R1 - originates on the Responder with IP 172.16.10.10 and has as destination the Initiator. This packet does not go through the RVS. As explained on section A.4.3, the RVS's only job is to relay the II packet. This obliges the Responder to be registered with the RVS so that the RVS may know the its current IP address. The detailed logs can be consulted in appendix B.3.
3.1.5 HIPL

Description

HIP for Linux - HIPL - is being developed under the InfraHIP project. InfraHIP assumes that the basic HIP protocol specification is almost complete, thus focussing on developing the missing infrastructure pieces such as DNS, NAT, and firewall support to enable a widespread deployment of HIP. InfraHIP is developed at the Helsinki Institute for Information Technology (HIIT) and has as industrial partners Nokia, Ericsson, Elisa and Finnish Defense Forces.

Operation

HIPL has a visible behaviour as described before. However the setup of the machine is different on all three implementations:

Interfaces: After starting the HIPL daemon (hipd/hipd), a dummy interface is automatically upped and configured with four inet6 addresses. They actually are the HITs attributed to the HIP host. These HITs are obtained based on four pairs of private/public keys.

```
hip_host_dsa_key_anon
hip_host_dsa_key_pub
hip_host_rsa_key_anon
hip_host_rsa_key_pub
hip_host_dsa_key_anon.pub
```
hip_host_dsa_key_pub.pub
hip_host_rsa_key_anon.pub
hip_host_rsa_key_pub.pub

These files are generated using "tools/hipconf" and should be saved on the /etc/hip/ directory.

The keys follow a naming structure:

1. The prefix of all keys is "hip_host"
2. The encryption/signing algorithm, "dsa" or "rsa"
3. The privacy type of the HIT, either "public" (can be e.g. published in dns) or "anon" (can be changed frequently)
4. There can be a ".pub" suffix:
   - If present, it represents the public part of the public/private key pair.
   - If not present, it represents the private part of the public/private key pair.

Mapping HITs to IPs: HIPL requires the explicit mapping of at least one HIT into one IP. This mapping is necessary so the HIPL daemon (hipd/hipd) knows what host has the IP and also make possible the communication through normal IP routes. The mapping can be done in two different ways for two different usages:

- For HIP aware applications
  The mapping between HIP and IPv6 is accomplished using the files /etc/hosts and /etc/hip/hosts (which needs to be created). In the /etc/hosts an entry must be appended mapping the IPv6 of the peer's physical interface with the peer's Hostname. It should look like:

  # IPv6 - HOSTNAME Mapping

In the /etc/hip/hosts file, it's necessary to make the mapping between the HIT and the peer's Hostname:

# HIT - HOSTNAME Mapping
1137:8397:73bc:924:ff26:f9fc:2f07:5dc6 HIPHostM1

The HIT used is one of the HITs automatically configured to the dummy device. The dummy device is only up when the daemon is running. The use of one of the 4 HITs attributed to the dummy will explicitly determine the combination of the RSA/DSA and pub/anon used.

This configuration can be tested using the test tool supplied and as explained at http://infrahip.hii.t.fi/hip/hipl/manual/ch06.html. test/conntest-client-gai is actually a HIP un-aware application, but is configured to resolve the HIT using the two files previously mentioned. This application creates 2 entries in the Security Policies Database (SPD), one inbound and another outbound.

Examples of Security Policies and Security Associations are presented ahead.
• For HIP un-aware applications
  The mapping between HIP and IPv6 is accomplished using the hipconf tool provided in the source code:

  tools/hipconf add map hit ipv6

  This tool must be used when the HIPL daemon (hipd/hipd) is running. The mapping will be automatically be deleted when the daemon is stopped. When the hipconf tool is used two SPD entries are created. These entries specify the HITS of both HIP Hosts.

Security Policies Database and Security Associations:
setkey -DP, which lists the Security Policies Database, will show something like:

created: Apr 5 11:57:14 2006 lastused: lifetime: 0(s) validtime: 0(s)
spid=8 seq=1 pid=9756 refcnt=1

created: Apr 5 11:57:14 2006 lastused: lifetime: 0(s) validtime: 0(s)
spid=9 seq=0 pid=9756 refcnt=1

The SAs will be formed after the first data packet is exchanged, which can be accomplished by simply pinging the peer's HIT.
setkey -D, that lists the active Security Associations, will show something like:

E: aes-cbc b08f3b26 4f16ec12 26338e78 98bd49f6 A: hmac-sha1 add3249 d61fa979 7a4c3e72 93c37f62 e07b593f
seq=0(0) flags=0(0) state=0
created: Apr 5 12:01:51 2006 current: Apr 5 12:02:00 2006
diff: 9(s) hard: 0(s) soft: 0(s)
last: Apr 5 12:01:51 2006 hard: 0(s) soft: 0(s)
current: 780(bytes) hard: 0(bytes) soft: 0(bytes)
allocated: 5 hard: 0 soft: 0
sadb_seq=1 pid=9822 refcnt=0

E: aes-cbc 3c334da8 d0c5a92c f682ff4f7 73b1f72a A: hmac-sha1 b6956f24 9a8f8fd2 0ce0c58f ba955b42 8e5e9854
seq=0(0) flags=0(0) state=mature
created: Apr 5 12:01:51 2006 current: Apr 5 12:02:00 2006
diff: 9(s) hard: 0(s) soft: 0(s)
last: Apr 5 12:01:51 2006 hard: 0(s) soft: 0(s)
current: 320(bytes) hard: 0(bytes) soft: 0(bytes)
allocated: 5 hard: 0 soft: 0
sadb_seq=0 pid=9822 refcnt=0


Conclusion
HIPL is a very complete solution for the implementation of HIP. It supports all of the basic processes like the base exchange, mobility, base exchange with RVS and mobility based on RVS. It also supports DNS integration and an experimental support for HIP3. By using the mapping tool provided (hipconf), hipl allows to choose which or the peer’s address do we wish to use. In this way we can explicitly choose to communicate with an anonymous or public HIT using the RSA or DSA algorithms. In a first phase of development, HIPL is, currently, a IPv6 only solution, i.e., only supports the use of HITS.

3.1.6 HIP4BSD
Description
HIP4BSD aims to develop a implementation of HIP for the BSD/Linux operating system. Although the main development is being made with particular focus on BSD, a Linux implementation is also available, but without so many capabilities. The implementation is being developed by the Ericsson Research in Finland at NomadicLab.

Operation
Configuration files: The main configuration file for HIP4BSD is hipconf.xml (see example in appendix C.1). This file contains the configuration parameters of the local HIP host (e.g., location of public and private key). It also contains the necessary parameters of the HIP peer necessary for the establishment of communication. The two mandatory parameters in this case are the remote HIT and the location on the local host of the file containing the peer’s public key.

The peer’s public key must be exchanged between the hosts - in the form of a file - and its location must be referenced in the hipconf.xml as said before. This file may look as follows:

```
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCGuBtdUA3GduewnamsI1IXg62QZq
ug+VJeQvSi7H2jYfAvBm4h10FEoa/Grpvbz9QJKjINZumEZvEBtNNyKfJ0yUe6Ak
hZWwgZ6HawtWhGZqI0GoWdavt50Bljp+C6tm2SNkMkjkvnD3rnZqjbpVMZsBG+k
xRSvIpUJKm4qA3AQIDQAQAB
-----END PUBLIC KEY-----
```

The hipconf.xml file, as referenced before, must contain the location of the public and private key of the local host: The public key is, as expected, similar to the presented above. The private key may look as follows:

```
-----BEGIN RSA PRIVATE KEY-----
MIICQIBAAKBgQDK2+5nuoETBLadZyiUvV4eGNZS/7b09gopg629rp9pe5M4u6C1u
```
Interfaces:  Like HIPL, a dummy device must be configured for the use of HIP4BSD. However in this case, this procedure must be done manually:

ifconfig dummy0 up

After the HIP4BSD daemon is running, this interface will be configured with an inet6 address which was defined on the configuration file hipconf.xml.

Mapping HITs to IPs:  No mapping between HITs and IPs is necessary. When the daemon is started, it automatically creates the necessary 1 entry in the SPD, which will look similar to the following:

created: Mar 28 16:48:59 2006 lastused: 1 lifetime: 0(s) validtime: 0(s)
spid=153 seq=0 pid=17155
refcnt=

In this example 1116:85bb:5890:4c6:ab86:acce:c3b4:1212 is the HIT of the local HIP host and
11a9:ac04:ff2e:64cb:d251:1c33:c435:ce5 is the HIT of the remote HIP peer.

Conclusion

HIP4BSD is the only implementation that contemplates the use of HIP on a BSD environment. Thus, should the BSD be chosen as developing environment, HIP4BSD would be the only possible choice. The linear implementation of HIP allowed for the most basic HIP testing like the Base Exchange and simple mobility. Its configuration is a bit complex but as the advantage of concentrating all of the parameters in one single file - hipconf.xml.

RVS testing was not possible, because RVS support is not available in the free HIP4BSD implementation.

3.1.7 OpenHIP

Description

OpenHIP is a free, open-source implementation of the Host Identity Protocol (HIP) which is being developed by Boeing.
Operation

Overview There are two ways to use OpenHIP, depending on your operating system and whether or not you want to patch your kernel:

1. With kernel support, for Linux only

2. Entirely in userspace, for Linux (user-mode HIP, or UMH) and Windows XP with Cygwin (HIP for Windows service or console app)

Both architectures consist of a user-space HIP daemon (hipd) and patches to IPSec tools. Also included is a hitgen utility used for initial setup and generating Host Identities, and scripts for setting up HIP.

1. Linux kernel support
   This mode provides a better performance, but requires the application of a patch to the kernel.

![Figure 3.4: OpenHIP - Kernel space implementation](image)

Linux applications connect to IP addresses and IPSec - checking the Security Policies Database - to see whether or not HIP is used. The ipsec-tools setkey utility, provided with the source, is used to set up a security policy requiring a HIP ESP tunnel for traffic destined for a specified peer. When packets are sent to that peer and match the policy entry, the kernel notifies hipd, hipd performs the HIP Base Exchange with the peer and sets up the Security Association using the native IPSec support from the 2.6 kernel. The resulting pair of SAs (incoming and outgoing) is associated with the HIT, so the underlying IP addresses may change without re-establishing the SAs.

2. Userspace Implementation
   The userspace implementation does not require any changes to the kernel, because all of the IPSec processing is done in a user process. A virtual device, the TAP driver\(^1\), is used to pick packets destined to LSIs and send them to the HIP

\(^1\)The TAP is a Virtual Ethernet network device. TAP driver was designed as low level kernel support for Ethernet tunneling. It provides userland application two interfaces: `/dev/tapX` - character device and `"tapX"` - virtual Ethernet interface
process. While this makes installation easier, the drawback is reduced performance as each user packet in a HIP association needs to be copied from user memory to the kernel, processed and sent back to the kernel to the real network interface.

![OpenHIP - User space implementation](image)

Figure 3.5: OpenHIP - User space implementation

Unlike the Linux version with kernel support, applications must connect using 32-bit LSIs to use HIP. The LSIs are of the form 1.x.x.x, where the last 24-bits are either the lower 24-bits of the HIT or some value specified in the identities file. A 1.0.0.0/8 route directs packets to the TUN/TAP\(^2\) driver, where they are sent to a listening socket in the HIP process. The HIP process features multiple threads for handling traffic input and output, and PFKEY and netlink functions that are normally services provided by the Linux kernel. This was designed in a manner such that the HIP daemon code (hipd) can be completely reused, running as a single thread in this HIP userspace process.

**Configuration Files**

The basic configuration file for OpenHIP is "hip.conf" that should be present in the "/etc/hip/". A example can be seen in appendix C.2. A basic hip.conf file is provided by default in the directory /hipd/conf, or can be generated by running "hitgen -conf".

HIP needs to store keys somewhere on the system. There are two types of keys that must be stored: your own Host Identities (there may be multiple on a system, including public and private key parts) and host identities of peer systems with whom you communicate. It is recommended that this values are stored in the my_host_identities.xml file which will contain public and private keys in use by hipd. This file should be kept in /etc/hip/my_host_identities.xml. An example can be seen in appendix C.3.

Finally, the known_host_identities.xml file, which stores the known HITs of other hosts that you may want to communicate with, and also any other tags needed to resolve the host identity ultimately to an IP address.

These tags include:

- \(<LSI>\) A local scoped identifier \(</LSI>\)

\(^2\)While TAP simulates an Ethernet device, TUN simulates a point-to-point network device.
• <name> A domain name </name>
• <addr> An IP address </addr>

An example of this example can be found on appendix C.4.

Conclusion

OpenHIP was the first HIP implementation tested in our project. It allowed to understand all of the components of the protocol. It provides a simple user space installation, which is ideal for early testing. For further testing and understanding of the protocols potential, it provides a kernel space installation with better performance and, surely, better suited for bigger testbeds. OpenHIP also provides a Windows implementation.

3.1.8 Conclusion about Implementations

The experiments with the implementations were highly conditioned by the release dates of some of the implementations modules. For example, after testing the Base Exchange and Mobility, with and without the use of RVS, with OpenHIP, the following step would be the integration of the test-bed with DNS. For that, it was necessary to use a modified DNS implementation. Unfortunately, the required patch that would modify the BIND DNS Server implementation wasn’t available at the time.

In the case of HIP4BSD, after the basic testing, we discovered that the use of HIP4BSD with RVS was not possible. The RVS feature isn’t available in the non-commercial version of HIP4BSD. Thus, the use of HIP4BSD was discarded.

HIPL is a very developed implementation that supports all of the basic functionalities, plus DNS integration and OpenDHT support. It has now experimental support for HI[3][2]. It has the disadvantage of being able to use HITs - LSIs are not supported yet.

3.2 HIP vs. MIP Comparison

HIP and MIP may, in some contexts, be direct competitors because of the similarity between the services and capabilities that each can provide. With this premise in mind, it became mandatory the elaboration of a comparison between them.

For our comparison, we’ll consider both behaviors of MIP - MIP through a Home Agent (MIP-HA) and MIP with Route Optimization (MIP-RO). This comparison was adapted from the work of Thomas R. Henderson[8].

The analysis can be made from different perspectives:

1. Performance
2. Security
3. Deployment
4. Scalability
5. Robustness

3OpenDHT is a publicly accessible Distributed Hash Table (DHT) service.
3.2.1 Performance

Important performance considerations are
- Per-session packet overhead and latency
- Per-packet bit overhead and latency
- Per-reallocations packet overhead and latency
- Computational load on end host and network infrastructure

<table>
<thead>
<tr>
<th>Technique</th>
<th>Performance Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIP-HA</td>
<td>- Suffers from suboptimal routing (all packets pass through the Home Agent)</td>
</tr>
<tr>
<td></td>
<td>- Requires per-packet encapsulation overhead</td>
</tr>
<tr>
<td></td>
<td>+ Micromobility extensions that improve fast handoff are being developed</td>
</tr>
<tr>
<td>MIP-RO</td>
<td>- Return Routability causes latency, overhead and computational load</td>
</tr>
<tr>
<td></td>
<td>+ Avoids sub-optimal routing of MIP-HA</td>
</tr>
<tr>
<td>HIP</td>
<td>- Requires operation and overhead of IPSec ESP</td>
</tr>
<tr>
<td></td>
<td>- Computational load, latency e overhead when performing the Base Exchange</td>
</tr>
<tr>
<td></td>
<td>- Requires additional network infrastructure to allow hosts to find mobile servers</td>
</tr>
</tbody>
</table>

Of all three techniques, MIP-HA has the worst performance in terms of packet route selection and additional per-packet overhead. MIP-RO requires a return routability check through the home network for each care-of address change, in addition to the binding update sent to the home network. If the return routability procedure is not required, and security associations (to set up a shared secret) can be reused across address changes, MIP-RO and HIPs readdressing performance should be similar. If IPSec were to be used with each of the protocols, HIP would have a small per-packet overhead advantage due to the avoidance of routing headers and destination options. MIPs micromobility extensions can improve the performance of the protocol.

3.2.2 Security

Mobility opens up the following potential security problems:
- Attacks against the mobility mechanism, e.g., replay attacks
- Host impersonation, e.g., man-in-the-middle attacks
- Privacy, i.e., the disclosure of the remote node's location

<table>
<thead>
<tr>
<th>Technique</th>
<th>Security Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIP-HA</td>
<td>+ Provides strong authentication and integrity of signaling packets</td>
</tr>
<tr>
<td></td>
<td>+ Compatible with IP security protocols</td>
</tr>
<tr>
<td></td>
<td>+ Privacy - node mobility can be hidden from the Correspondent Nodes</td>
</tr>
<tr>
<td></td>
<td>+ Can use IPSec extensions</td>
</tr>
<tr>
<td>MIP-RO</td>
<td>- No privacy</td>
</tr>
<tr>
<td></td>
<td>- Without the Return Routability procedure inherits the same problems of MIP-HA</td>
</tr>
<tr>
<td></td>
<td>+ Using the Return Routability procedure secures the binding updates to CNs</td>
</tr>
<tr>
<td>HIP</td>
<td>- No privacy - HIP privacy extensions are being considered</td>
</tr>
<tr>
<td></td>
<td>- Uses IPSec by default</td>
</tr>
<tr>
<td></td>
<td>- Implements &quot;stateless&quot; connection handshake for Denial-of-Service resilience</td>
</tr>
</tbody>
</table>
Security and mobility management are inexorably intertwined, because mobility opens up the potential for a number of security problems, including attacks against the mechanisms of mobility host impersonation and privacy.

All approaches fundamentally require some kind of trust relationship between the mobile node and some network infrastructure (the home agents in MIP, RVS in HIP), but it is plausible to expect that such relationships could be provisioned. However, the challenge for all the approaches is to avoid needing a preconfigured trust relationship with all possible correspondent nodes. All approaches are basically compatible with IPsec as a way to protect data content. MIP requires the use of an extension to the protocol and HIP uses it by default.

### 3.2.3 Deployment

Deployment issues include:

- Required changes to end hosts, network infrastructure and applications
- Ability to coexist with middleboxes, e.g., NAT or proxy
- Potential for auto-configuration
- Reliance on non-ubiquitous services, e.g., anycast and multicast
- IPv4 vs. IPv6 issues

<table>
<thead>
<tr>
<th>Technique</th>
<th>Deployment Issues</th>
</tr>
</thead>
</table>
| MIP-HA    | - Requires deployment and administration of Home Agents  
           | - Operation is challenged by presence of firewalls and NAT  
           | + Requires no changes to Correspondent Nodes or DNS |
| MIP-RO    | - May require a Return Routability extension  
           | - All of MIP-HA issues |
| HIP       | - Requires IPsec deployment  
           | - Requires extensions to DNS, for practical use  
           | - Requires changes to the network stack on both end hosts - HIP aware hosts  
           | - Requires the presence of a RVS, for practical use |

Deployment of new services is increasingly complicated by backward compatibility concerns and the growth of functionality in the network that can inadvertently or intentionally block the operation of protocols.

MIP-HA has a significant deployment advantage over the other proposals in that it does not require changes to correspondent nodes, and has been standardized for some time and is commercially available. However, as mentioned above, MIP deployment has been hindered by the presence of firewalls and NAT. NAT in particular poses a serious problem to the MIP architecture because many hosts may not have publicly routable home or care-of addresses. It has long been thought that a transition to IPv6 would solve this situation, but as IPv6 transition becomes less clear, and unless the IPv6 architecture has a compelling argument against the continued use of NAT, such problems may persist.

Deployment issues are most severe for a proposal such as HIP, which requires widespread deployment of IPsec, more radical changes to the networking stacks of both connection endpoints, alterations to DNS and the deployment of another physical infrastructure - RVS.
3.2.4 Scalability

<table>
<thead>
<tr>
<th>Technique</th>
<th>Scalability Issues</th>
</tr>
</thead>
</table>
| MIP-HA    | - Uses wide distribution of home agents (in each home network) to handle the many mobility bindings  
+ Per-packet operations could be implemented in hardware, reducing expensive overhead  
+ Micromobility techniques to reduce the scope of frequent binding updates |
| MIP-RO    | - More scalable than MIP-HA in terms of bandwidth usage and avoiding home network bottlenecks |
| HIP       | - The use of DNS for HIT resolution causes scalability problems  
- The possible implementation of RVS using DHT server could improve scalability |

Traditionally, scalability in the Internet is achieved through hierarchy and by moving functionality out of the network to the end points.

MIP scales by widely distributing the infrastructure - Home Agents - required for its operation. In the case of MIP-RO, the scalability is improved by reducing the overhead produced by sub-optimal routing. HIP scales by adopting an end-to-end solution. Currently, IP addresses are often aggregated for reduced administration purposes. The use of Host Identities with structure and without aggregation based on network prefix can cause scalability problems.

3.2.5 Robustness

Robustness issues include:

- Fault tolerance
- Ability to handle simultaneous host mobility
- Potential for auto-configuration
- Ability to handle irregularities in routing, e.g., unidirectional links
- Multihoming capability

<table>
<thead>
<tr>
<th>Technique</th>
<th>Robustness Issues</th>
</tr>
</thead>
</table>
| MIP-HA    | - Home Agents or home networks can constitute single points of failure  
+ Per-packet operations could be implemented in hardware, reducing expensive overhead  
+ Micromobility techniques to reduce the scope of frequent binding updates |
| MIP-RO    | - More scalable than MIP-HA in terms of bandwidth usage and avoiding home network bottlenecks |
| HIP       | - The use of DNS for HIT resolution causes scalability problems  
- The possible implementation of RVS using DHT server could improve scalability |

All of the techniques are vulnerable to disconnection from the home network, whether through a Home Agent or a RVS. Any mobile node is oblivious to failures
in the infrastructure until a re-registration occurs. In MIP-HA the deployment of redundant Home Agents may diminish this problem. The same approach can be done by HIP, deploying more than one RVS. Another approach for HIP is the implementation of RVSs in DHT servers, hence eliminating single points of failure. MIP-RO is more independent of the Home Agent as it simply uses it immediately after a mobility procedure.

3.2.6 Summary

- **Performance**
  - All techniques create additional overhead, latency and computational load.
  - MIP-HA has the worst performance.
  - MIP-RO and HIP have similar performance.

- **Security**
  - MIP-HA has good security, supporting all the IP security protocols.
  - MIP-RO requires a security mechanism extension - Return Routability.
  - HIP has embedded security using IPSec.

- **Deployment**
  - MIP requires the deployment of Home Agents.
  - HIP is a disruptive technology that requires alterations to the host’s network stack, the introduction of another item in the networking infrastructure - RVS and alterations to DNS.

- **Scalability**
  - MIP-RO is a good solution with possibilities of evolution through the integration of micromobility extensions.
  - Using DNS for HIT resolution causes scalability problems.

- **Robustness**
  - MIP creates single points of failure if uses non-redundant Home Agents
  - HIP’s distributed RVS - using DHT - would improve robustness

3.2.7 HIP vs. MIP Conclusions

In the comparison tables and in the summary, we presented how do the protocols match to its opponents. The conclusions are obvious: MIP is a more complete solution but maybe because of the head start it has over HIP. HIP offers clear operational advantages over MIP-HA but not when compared to MIP-RO.

The success of either protocol deeply depends on the evolution of the Internet. If IPv6 becomes widely accepted in the future, it will be very difficult to beat the operational capabilities of MIP-RO. However, if a necessity for decoupled development of network and transport layers arises, and the introduction of a namespace is required, HIP may be an alternative.
3.2.8 Conclusions

In our work with HIP, we were able to experiment with all of the available implementations. These experiments produced a very solid knowledge about the protocol and its capabilities and the advantages and disadvantages of each implementation.

HIP is clearly in its first steps, with many problems still without resolution. During the time of the project, further developments were constantly being done to the protocol and to the implementations and it was noticeable that the more the each evolved, more problems came to light. It is a continuous learning experience. This experience produced a HIP Manual which is a compilation of the information gathered during the project.

A recurring subject during this project was "What does HIP compare to MIP?". This caused us to elaborate a HIP vs. MIP comparison which allowed to understand both protocol's capabilities.

3.3 SIP Application

3.3.1 Objectives

This part of the work consisted on the development of an application that would use the Session Initiation Protocol to implement application level mobility. The application should allow:

- The establishment of a SIP session between a media client and a media server
- The server should transmit a media stream
- The client should display that media stream
- The client should be able to trigger a mobility procedure in which a ongoing session would be transferred to another terminal
- The client should be able to terminate the sip session

Also, the development of this application allowed for an enriching learning experience with Java and Software Engineering techniques.

3.3.2 Motivation

To develop a combined mobility solution, it should allow mobility between networks and between terminals. The first problem was addressed by HIP. The solution for the second problem had to come from an upper stack layer, since TCP doesn't allow the dynamic change of the end-points of connections. The correct approach here is, from our point of view, to develop a solution at the upper layers (session layer or at the application layer). That said, SIP was an obvious candidate. It is a session management protocol which allows us to establish and finish sessions, negotiate session parameters, and notify the session peers about events.

3.3.3 Description of the Application

The description of the application will be done as suggested by the good-practices of software engineering.
Use Cases Overview

The application will comprise 3 use cases:

- **startSession**: The user, by interacting with a client GUI, will be able to start a session with a remote media server. The user must insert in a text field the address of the media server, the local port for the reception of traffic and the desired file.

- **transferSession**: The user, by interacting with a client GUI, will be able to transfer the ongoing session to another terminal, which will be running another instance of the client.

- **endSession**: The user, by interacting with a client GUI, will be able to terminate the session.

The use cases will be detailed ahead under the Use Cases - Detail subsection.

Actors

The actors represent the agents that interact with a system. In our case, the only actor in the use cases is the User. The User may act freely with the GUI.

Domain Model

Representation of the the relations between classes.
Figure 3.7: Domain Model

- **sipClient**
  Implements all of the SIP signaling, on the client side of the application. This class also launches a thread of the class which will receive the media stream from the server - AVReceive.

- **sipServer**
  Implements all of the SIP signaling on the server side of the application. This class also launches a thread of the class which will send the media stream - AVTransmit.

- **AVReceive**
  Class implemented on the client machine, responsible for the reception of the media stream.

- **AVTransmit**
  Class implemented on the server machine, responsible for the transmission of the media stream.

- **clientAppUI**
  Class responsible for the implementation of the GUI which allows the User to
interact with the application.

- **MyRouter**
  This class implements the Router interface. The Router interface defines methods that allow the obtention the default route and the outbound proxy. The default route and the outbound proxy can be constituted by several hops, *i.e.*, network nodes.

- **HopImpl**
  This class implements the Hop interface. It allows to obtain the next route hop, returning its address, port and transport protocol.

The Javadocs of theses classes can be found at

http://paginas.fe.up.pt/~ee04238/ES/javadoc/

or on the CD accompanying this document.

**Logic Architecture**

The application is based on the server-client paradigm. Thus, the logic architecture will represent two different logical implementations:

- **Client**

![Diagram of Client's Logic Architecture](image)

Figure 3.8: Client's Logic Architecture

- GUI - Object that contains the graphical interface which will allow the User to interact with the client application.
- Client Package - This package contains all of the classes responsible for SIP signaling and reception of the media stream. The class sipClient uses the JAIN-SIP Framework to implement all of the SIP signaling necessary to the establishment of a session, client-side. The class AVRReceive uses the Java Media Framework - JMF - to implement all of the classes necessary for the reception and treatment of the media stream.

- JAIN-SIP - JAIN is an initiative that aims to define a group of Java APIs that will allow a fast development of next generation communication technologies. One of these APIs is JAIN-SIP which implements a set of procedures that follow the SIP norms. This way, this API assures the correct use of SIP.

- The Java Media Framework (JMF) API provides ways to add audio, video or other media types to Java based applications or applets. The JMF allows capturing, playback and streaming of various types of media.

- Server

![Figure 3.9: Servers's Logic Architecture](image)

- Server Package - This package contains all of the classes responsible for server-side SIP signaling and transmission of a media stream. The class sipServer uses the JAIN-SIP Framework to implement all of the server-side SIP signaling. The class AVTransmit uses JMF to implement all of the classes necessary for the transmission of the stream.

- JAIN-SIP and JMF as defined for the Client.

### Physical Architecture

The physical architecture describes how the different application components will be distributed. In this case, being an application based on the server-client paradigm, the application will be distributed differently by two machines.

The clientApp component contains:
• GUI
• sipClient
• AVReceive

The serverApp component contains:
• sipServer
• AVTransmit

JAIN-SIP provides a number of libraries necessary to the implementation. The sipClient and sipServer classes depend on:
• JainSipApi1.1.jar
• nist-sdp-1.0.jar
• nist-sip-1.2.jar
• sip-sdp.jar

sipClient and sipServer also need some other libraries:
• jakarta-regexp-1.3.jar
• junit.jar
• log4j-1.2.8.jar

The JMF provides the AVReceive and AVTransmit classes with jmf.jar library.

The installation of JMF is detailed in appendix D.1.

Use Cases - Detail

The use cases, i.e., the interaction of the User with the application is done by using a Graphical User Interface - GUI.
**startSession**
This use case implements the exchange of messages and content as depicted in fig. 3.12.

The behavior of the application is shown on fig. 3.13

- **Call sipClient**: By action of the User, the GUI call the methods of the sipClient class that will trigger the SIP signaling procedure.
- **SIP Signaling**: The sipClient and sipServer classes initiate a SIP transaction.
- **Call AVRReceive & Call AVTransmit**: When the transaction is terminated, as specified in fig. 3.12, sipClient and sipServer call AVRReceive and AVTransmit respectively, initiating the transmission and reception procedures.
- **Media Stream**: Transmission of the stream from AVTransmit to AVRReceive.

**transferSession**
This use case implements the exchange of messages and content as depicted in fig. 3.14.

The SIP session and media transmission are initiated as explained before for the startSession use case. The messages related with the mobility procedure are explained next.

- **Call sipClient**: The user acts upon the GUI, starting the application in passive mode. In this mode, the application waits for a REFER message.
Figure 3.12: startSession messages (1) - ACK sent by sipClient running on the Client machine. After the ACK, the class sipClient launches an instance of AVRReceive as a thread. This thread will wait for a media stream. (2) - The ACK is received by the class sipServer running on the Server machine. After the reception, sipServer launches an instance of AVTransmit, as a thread that will send the media stream to the Client.

- **REFER**: By action of the User, the GUI call the methods of the sipClient class that will trigger transfer of the media session to another client (Client 2). This message is sent to the server, to inform it that an alteration to the session will occur and to the other client. The REFER message sent to the server will have a Refer-To: header which will indicate the address of Client 2. The REFER sent to Client 2 will have a Refer-To: header which will contain the address of the server to which the INVITE should be sent. This message will only be sent after the Server accepts the REFER message with a 202 (Accepted) response, as shown in fig. 3.14.

- **INVITE**: The class sipClient at Client 2 sends an INVITE message to the Server. The Server will be in a waiting state since the reception of the REFER message from Client 1.

- **Call**: The sipServer class, after terminating the SIP negotiation, triggered by the INVITE message from Client 2, will terminate the instance of AVTransmit that was transmitting to Client 1 (Stream End) and will start another instance of AVTransmit, that will start to transmit to Client 2 (New Media Stream).

- **Call AVRReceive**: After the SIP negotiation with Server is terminated, the sipClient class at Client 2 will launch an instance of AVRReceive and will wait for the New Media Stream to arrive.

dinandSession
This use case is used to terminate the the SIP session and the media stream. The message exchange occurs as depicted in fig. 3.16.

### 3.3.4 Results
For testing of the application, we isolated the three use cases. The implementation of each use case can be seen by the captured message logs.
Figure 3.13: startSession behaviour

**startSession Use Case**

As shown in the use cases detailed description, this use case is responsible for the initiation of the SIP session and the reception of the media stream, as seen on fig.3.12. The operation of this use case is depicted in the fig. 3.17.

As it can be seen in the fig.3.17, the Client sends an INVITE message to the Server. The INVITE message is repeated until the Server responds or until a timeout occurs. The Server responds with a 180 (Ringing) message, alerting the Client that the INVITE request is being treated. The Client then, sends a 200 (OK) response, indicating the Client that the INVITE request was accepted. The Client confirms with an ACK message.

After the SIP signaling procedure is complete, the UDP transaction begins. When the user defines a port using the GUI, that port will be used for the reception of one type of media. Because the requested file was a audio and video file two different ports appear in the log. It is imposed that the user’s designated port must be an even number. In our case, port 12344. This is because the following odd number will be used as a control port. Thus, in our case, we have:

- Port 12344 - Designated port for the reception of the first media type, e.g., audio.
- Port 12345 - Control port for first media type.
- Port 12346 - Designated port for the reception of the second media type, e.g., video.
- Port 12347 - Control port for second media type.

Because the Client actually receives two media flows, typically one audio and one video, the application opens two different player windows. As this work is purely experimental, no effort was done to present the media to the user in a more pleasant way. The player looks as seen in fig.3.18
Figure 3.14: transferSession messages (1) - ACK sent by sipClient running on the Client machine. After the ACK, the class sipClient launches an instance of AVReceive as a thread. This thread will wait for a media stream. (2) - The ACK is received by the class sipServer running on the Server machine. After the reception, sipServer launches an instance of AVTransmit, as a thread that will send the media stream to the Client. (3) - The User uses the GUI to trigger the transfer of the session.

**transferSession Use Case**

This use case would be responsible for transferring the media stream to another Client machine. At the time of the elaboration of this document, this use case wasn’t completely implemented. As shown on fig. 3.14, the Client should inform the server of the new address to which the stream should be sent. This was accomplished by using a REFER message. The Server should then wait to receive an invite from a second client, whose address must be the one referenced by the first client. That was also accomplished. Finally the first Client should inform the second Client of the Server’s address and also provide information about the stream currently being received. This functionality was not implemented.

However, to test the implemented functionalities of this use case, some alterations were done to the second Client application. The log captured is depicted in fig.3.19. As it can be seen in the figure, after the session is established, a REFER message is sent to the Server (defined in the To: header) which had the 172.16.0.12 IP address.
This message is responded by the first 202 (Accepted) message. The second REFER message and consequent 202 response should be delivered to the second Client, which would trigger an INVITE from it to the Server. This second REFER message should have a SDP (Session Description Protocol) message body in which the information about the ongoing stream would be carried. As said before, this message was not implemented.

After the alterations to the client application running on the second Client machine, we could simulate the behaviour of the Server as if this use case was correctly implemented. The Server’s behaviour can be seen on fig. 3.20.

As it can be seen in the fig. 3.20, After the SIP session is established with the first Client, a REFER request is received with 172.16.0.200 as origin address (first Client). As referred above, the second REFER should not have been sent to the Server. The REFER request clearly indicates Request: REFER sip:Refer. sip:Refer is the second Client’s SIP URI and tell the Server that should expect an INVITE from a Client whose identification must be sip:Refer.

After the REFER is accepted by the Server, the Server waits, stateless, for an INVITE to arrive. After receiving and verifying the INVITE shown on the figure, the normal SIP negotiation proceeds.

deSession Use Case

This use case simply terminates the SIP session and orders the termination of the JMF threads launched. A log of this use case can be seen in fig. 3.21. In the figure we can observe that, after a session is established, a BYE request is sent from the Client to the Server. The latter responds with a 200 (OK) response.

3.3.5 Conclusions about the application

The proposed application should comprise the three use cases as described above. The startSession use case was implemented successfully. A user can use the GUI to request a certain file that, a few seconds later starts to play in the user’s machine. This use case uses all of the technology involved in the application - SIP signaling and media handling with the JMF.

The transferSession use case was not implemented successfully. This use case required some alterations to the usual behaviour of SIP. We proposed the use of the REFER extension to the SIP protocol for application mobility, i.e., the transference of a ongoing session - a data flow - to another terminal. At the time of the elaboration of this document, the use case was not completely implemented and its practical results were not visible.

The endSession use case was implemented with success. The user, pressing the End button on the GUI would trigger the termination of the application.
3.4 Integrated operation of the testbed with the application

The integration the HIP testbed with SIP is accomplished by running the SIP application on the machines running a HIP implementation. The application should be configured with the HIP addresses thus forcing the use of HIP. The application was developed using IPv4 type addresses, so, for the integrated tests, LSI must be uses instead of HITs. Further development of the application would permit the ability to use either LSIs or HITs.

3.4.1 Results

After the HIP Base exchange, all of the data exchanged between the hosts is encapsulated using ESP. So, in the captured log, we can only see the HIP base exchange and the ESP data flow that is exchanged after. The log is shown on fig.3.22.

The Session Mobility scenario in HIP - changing the network attachment point, as explained in is shown in fig.3.23. Once again, ESP encapsulates the data flow. As seen in the figure, the Server with the address 172.16.0.102 is sending data to the Client with the address 172.16.0.147. Then, the Client initiates the mobility procedure. After acquiring a different IP address, it informs the Server with a UPDATE message. The mobility procedure happens as explained in A.4.5. After the three UPDATE messages are exchanged, the stream continues from the Server to the Client at his new address.

Observing the timestamps, we see that there's a large period of time without receiving any stream - around 21 seconds. From our experiments, this value was very variable. As it can be seen also in the timestamps, the major portion of this time is until the Client acquires a new IP address. After the first UPDATE message is sent, the data stream is resumed approximately after 1 second.
Figure 3.15: transferSession behaviour
Figure 3.16: endSession messages (1) - ACK sent by sipClient running on the Client machine. After the ACK, the class sipClient launches an instance of AVReceive as a thread. This thread will wait for a media stream. (2) - The ACK is received by the class sipServer running on the Server machine. After the reception, sipServer launches an instance of AVTransmit, as a thread that will send the media stream to the Client.

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>09:00-09:07-01</td>
<td>127.16.0.200</td>
<td>127.16.0.12</td>
<td>SIP/UDP</td>
<td>INVITE sip:server/with session description</td>
</tr>
<tr>
<td>26</td>
<td>09:00-09:07-01</td>
<td>127.16.0.12</td>
<td>127.16.0.200</td>
<td>SIP/UDP</td>
<td>Status: 200 OK</td>
</tr>
<tr>
<td>27</td>
<td>09:00-09:07-01</td>
<td>127.16.0.12</td>
<td>127.16.0.200</td>
<td>SIP/UDP</td>
<td>Status: 200 OK</td>
</tr>
<tr>
<td>34</td>
<td>09:00-09:07-01</td>
<td>127.16.0.12</td>
<td>127.16.0.200</td>
<td>UDP</td>
<td>Source port: 12346 Destination port: 12345</td>
</tr>
<tr>
<td>37</td>
<td>09:00-09:07-01</td>
<td>127.16.0.12</td>
<td>127.16.0.200</td>
<td>UDP</td>
<td>Source port: 12345 Destination port: 12345</td>
</tr>
<tr>
<td>40</td>
<td>09:00-09:07-01</td>
<td>127.16.0.12</td>
<td>127.16.0.200</td>
<td>UDP</td>
<td>Source port: 12346 Destination port: 12346</td>
</tr>
</tbody>
</table>

From 23 (77 bytes on wire, 71 bytes captured)
Ethernet II, Src: 00:21:01:0c:00:00, Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
Internet Protocol, Src: 127.16.0.200 (127.16.0.200), Dst: 127.16.0.12 (127.16.0.12)
User Datagram Protocol, Src Port: 50006, Dst Port: 5079 (5079)
Session Initiation Protocol
- Present-From: INVITE sip:server SIP/2.0
- Call-ID: endSession:0716:800056ebd193b580f6f3672.16.0.200
- CSeq: 1 INVITE
- From: "Client Name" sip:client:tag=12345
- To: "Server" sip:server/sip:server
- Via: SIP/2.0/UDP 127.16.0.200;branch=v00000000;branch=v00000000;branch=v00000000;branch=v00000000
- Max-Forwards: 70
- Contact: "Client Name" sip:client:tag=12345
- My-redirect: my new header value
- Content-Type: application/sdp
- My-Other-Header: my own header value
- Call-ID: http://www.example.com
- Content-Length: 262

Figure 3.17: startSession Ethereal log
Figure 3.18: Video and Audio players

<table>
<thead>
<tr>
<th>No</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>174 2004-07-02 15:12:49 818049</td>
<td>172.16.0.12</td>
<td>SIP/SDP Request: INVITE sip:server, with session description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>174 2004-07-02 15:12:49 818049</td>
<td>172.16.0.12</td>
<td>SIP/SDP Request: INVITE sip:server, with session description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>178 2004-07-02 15:12:49 818049</td>
<td>172.16.0.12</td>
<td>SIP/SDP Request: INVITE sip:server, with session description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>178 2004-07-02 15:12:49 818049</td>
<td>172.16.0.12</td>
<td>SIP/SDP Request: INVITE sip:server, with session description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>180 2004-07-02 15:12:49 818049</td>
<td>172.16.0.12</td>
<td>SIP/SDP Request: INVITE sip:server, with session description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>180 2004-07-02 15:12:49 818049</td>
<td>172.16.0.12</td>
<td>SIP/SDP Request: INVITE sip:server, with session description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>181 2004-07-02 15:12:49 818049</td>
<td>172.16.0.12</td>
<td>SIP/SDP Request: INVITE sip:server, with session description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>181 2004-07-02 15:12:49 818049</td>
<td>172.16.0.12</td>
<td>SIP/SDP Request: INVITE sip:server, with session description</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.19: transferSession Ethereal log as seen on the first Client machine
### transferSession Ethereal log as seen on the Server machine

#### Figure 3.20:
```
<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2006-07-01 09:58:27</td>
<td>172.16.0.200</td>
<td>172.16.0.12</td>
<td>SIP</td>
<td>Error: INVITE sip:Server</td>
</tr>
<tr>
<td>2</td>
<td>2006-07-01 09:58:27</td>
<td>172.16.0.200</td>
<td>172.16.0.12</td>
<td>SIP</td>
<td>Error: INVITE sip:Server</td>
</tr>
<tr>
<td>3</td>
<td>2006-07-01 09:58:27</td>
<td>172.16.0.200</td>
<td>172.16.0.12</td>
<td>SIP</td>
<td>Error: INVITE sip:Server</td>
</tr>
<tr>
<td>4</td>
<td>2006-07-01 09:58:27</td>
<td>172.16.0.200</td>
<td>172.16.0.12</td>
<td>SIP</td>
<td>Error: INVITE sip:Server</td>
</tr>
<tr>
<td>5</td>
<td>2006-07-01 09:58:27</td>
<td>172.16.0.200</td>
<td>172.16.0.12</td>
<td>SIP</td>
<td>Error: INVITE sip:Server</td>
</tr>
<tr>
<td>6</td>
<td>2006-07-01 09:58:27</td>
<td>172.16.0.200</td>
<td>172.16.0.12</td>
<td>SIP</td>
<td>Error: INVITE sip:Server</td>
</tr>
<tr>
<td>7</td>
<td>2006-07-01 09:58:27</td>
<td>172.16.0.200</td>
<td>172.16.0.12</td>
<td>SIP</td>
<td>Error: INVITE sip:Server</td>
</tr>
<tr>
<td>8</td>
<td>2006-07-01 09:58:27</td>
<td>172.16.0.200</td>
<td>172.16.0.12</td>
<td>SIP</td>
<td>Error: INVITE sip:Server</td>
</tr>
<tr>
<td>9</td>
<td>2006-07-01 09:58:27</td>
<td>172.16.0.200</td>
<td>172.16.0.12</td>
<td>SIP</td>
<td>Error: INVITE sip:Server</td>
</tr>
<tr>
<td>10</td>
<td>2006-07-01 09:58:27</td>
<td>172.16.0.200</td>
<td>172.16.0.12</td>
<td>SIP</td>
<td>Error: INVITE sip:Server</td>
</tr>
<tr>
<td>12</td>
<td>2006-07-01 09:58:27</td>
<td>172.16.0.200</td>
<td>172.16.0.12</td>
<td>SIP</td>
<td>Error: INVITE sip:Server</td>
</tr>
<tr>
<td></td>
<td>Total: 14</td>
<td></td>
<td></td>
<td></td>
<td>Error: INVITE sip:Server</td>
</tr>
</tbody>
</table>
```

### endSession Ethereal log

#### Figure 3.21:
```
<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2006-07-01 09:58:27</td>
<td>172.16.0.200</td>
<td>172.16.0.12</td>
<td>SIP</td>
<td>Error: INVITE sip:Server</td>
</tr>
<tr>
<td>2</td>
<td>2006-07-01 09:58:27</td>
<td>172.16.0.200</td>
<td>172.16.0.12</td>
<td>SIP</td>
<td>Error: INVITE sip:Server</td>
</tr>
<tr>
<td>3</td>
<td>2006-07-01 09:58:27</td>
<td>172.16.0.200</td>
<td>172.16.0.12</td>
<td>SIP</td>
<td>Error: INVITE sip:Server</td>
</tr>
<tr>
<td>4</td>
<td>2006-07-01 09:58:27</td>
<td>172.16.0.200</td>
<td>172.16.0.12</td>
<td>SIP</td>
<td>Error: INVITE sip:Server</td>
</tr>
<tr>
<td>5</td>
<td>2006-07-01 09:58:27</td>
<td>172.16.0.200</td>
<td>172.16.0.12</td>
<td>SIP</td>
<td>Error: INVITE sip:Server</td>
</tr>
<tr>
<td>6</td>
<td>2006-07-01 09:58:27</td>
<td>172.16.0.200</td>
<td>172.16.0.12</td>
<td>SIP</td>
<td>Error: INVITE sip:Server</td>
</tr>
<tr>
<td>7</td>
<td>2006-07-01 09:58:27</td>
<td>172.16.0.200</td>
<td>172.16.0.12</td>
<td>SIP</td>
<td>Error: INVITE sip:Server</td>
</tr>
<tr>
<td>8</td>
<td>2006-07-01 09:58:27</td>
<td>172.16.0.200</td>
<td>172.16.0.12</td>
<td>SIP</td>
<td>Error: INVITE sip:Server</td>
</tr>
<tr>
<td>9</td>
<td>2006-07-01 09:58:27</td>
<td>172.16.0.200</td>
<td>172.16.0.12</td>
<td>SIP</td>
<td>Error: INVITE sip:Server</td>
</tr>
<tr>
<td>10</td>
<td>2006-07-01 09:58:27</td>
<td>172.16.0.200</td>
<td>172.16.0.12</td>
<td>SIP</td>
<td>Error: INVITE sip:Server</td>
</tr>
<tr>
<td>12</td>
<td>2006-07-01 09:58:27</td>
<td>172.16.0.200</td>
<td>172.16.0.12</td>
<td>SIP</td>
<td>Error: INVITE sip:Server</td>
</tr>
<tr>
<td></td>
<td>Total: 14</td>
<td></td>
<td></td>
<td></td>
<td>Error: INVITE sip:Server</td>
</tr>
</tbody>
</table>
```

```c
```
```
<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol info</th>
</tr>
</thead>
<tbody>
<tr>
<td>289</td>
<td>2006-03-09 12:10:04.648092</td>
<td>172.16.0.147</td>
<td>172.16.0.102</td>
<td>HIP (HIP Initiator Packet)</td>
</tr>
<tr>
<td>290</td>
<td>2006-03-09 12:10:04.648105</td>
<td>172.16.0.102</td>
<td>172.16.0.147</td>
<td>HIP (HIP Responder Packet)</td>
</tr>
<tr>
<td>291</td>
<td>2006-03-09 12:10:04.749905</td>
<td>172.16.0.147</td>
<td>172.16.0.102</td>
<td>HIP (Second HIP Initiator Packet)</td>
</tr>
<tr>
<td>292</td>
<td>2006-03-09 12:10:04.750277</td>
<td>172.16.0.102</td>
<td>172.16.0.147</td>
<td>HIP (Second HIP Responder Packet)</td>
</tr>
<tr>
<td>301</td>
<td>2006-03-09 12:10:09.872497</td>
<td>172.16.0.102</td>
<td>172.16.0.147</td>
<td>ESP (ESP with EAPv0)</td>
</tr>
<tr>
<td>304</td>
<td>2006-03-09 12:10:09.877105</td>
<td>172.16.0.147</td>
<td>172.16.0.102</td>
<td>ESP (ESP with EAPv2)</td>
</tr>
<tr>
<td>305</td>
<td>2006-03-09 12:10:09.877302</td>
<td>172.16.0.102</td>
<td>172.16.0.147</td>
<td>ESP (ESP with EAPv0)</td>
</tr>
</tbody>
</table>

*p Frame 302 (50 bytes on wire, 150 bytes captured) *
*Ethernet II, Src: 00:11:0c:20:4e:32, Dst: 00:11:0c:29:4e:32, Len: 40 (40 bytes) *
*Frame 302 (50 bytes on wire, 150 bytes captured) *
*Ethernet II, Src: 00:11:0c:20:4e:32, Dst: 00:11:0c:29:4e:32, Len: 40 (40 bytes) *
*Frame 302 (50 bytes on wire, 150 bytes captured) *
*Ethernet II, Src: 00:11:0c:20:4e:32, Dst: 00:11:0c:29:4e:32, Len: 40 (40 bytes) *
*Frame 302 (50 bytes on wire, 150 bytes captured) *
*Ethernet II, Src: 00:11:0c:20:4e:32, Dst: 00:11:0c:29:4e:32, Len: 40 (40 bytes) *
*Frame 302 (50 bytes on wire, 150 bytes captured) *
*Ethernet II, Src: 00:11:0c:20:4e:32, Dst: 00:11:0c:29:4e:32, Len: 40 (40 bytes) *
*Frame 302 (50 bytes on wire, 150 bytes captured) *
*Ethernet II, Src: 00:11:0c:20:4e:32, Dst: 00:11:0c:29:4e:32, Len: 40 (40 bytes) *
*Frame 302 (50 bytes on wire, 150 bytes captured) *
*Ethernet II, Src: 00:11:0c:20:4e:32, Dst: 00:11:0c:29:4e:32, Len: 40 (40 bytes) *
*Frame 302 (50 bytes on wire, 150 bytes captured) *
*Ethernet II, Src: 00:11:0c:20:4e:32, Dst: 00:11:0c:29:4e:32, Len: 40 (40 bytes) *
*Frame 302 (50 bytes on wire, 150 bytes captured) *
*Ethernet II, Src: 00:11:0c:20:4e:32, Dst: 00:11:0c:29:4e:32, Len: 40 (40 bytes) *
*Frame 302 (50 bytes on wire, 150 bytes captured) *
*Ethernet II, Src: 00:11:0c:20:4e:32, Dst: 00:11:0c:29:4e:32, Len: 40 (40 bytes) *

Figure 3.22: Integrated operation of HIP and SIP

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol info</th>
</tr>
</thead>
<tbody>
<tr>
<td>348</td>
<td>2006-03-09 12:10:18 875656</td>
<td>172.16.0.102</td>
<td>172.16.0.147</td>
<td>ESP (ESP with HIPv3)</td>
</tr>
<tr>
<td>372</td>
<td>2006-03-09 12:10:30 327728</td>
<td>172.16.0.102</td>
<td>172.16.0.147</td>
<td>HIP UPDATE (HIP Update Packet)</td>
</tr>
<tr>
<td>373</td>
<td>2006-03-09 12:10:30 327728</td>
<td>172.16.0.102</td>
<td>172.16.0.147</td>
<td>HIP UPDATE (HIP Update Packet)</td>
</tr>
<tr>
<td>374</td>
<td>2006-03-09 12:10:30 327728</td>
<td>172.16.0.102</td>
<td>172.16.0.147</td>
<td>HIP UPDATE (HIP Update Packet)</td>
</tr>
</tbody>
</table>

*p Frame 371 (260 bytes on wire, 260 bytes captured) *
*Ethernet II, Src: 00:11:0c:20:4e:32, Dst: 00:11:0c:29:4e:32, Len: 40 (40 bytes) *
*Frame 371 (260 bytes on wire, 260 bytes captured) *
*Ethernet II, Src: 00:11:0c:20:4e:32, Dst: 00:11:0c:29:4e:32, Len: 40 (40 bytes) *
*Frame 371 (260 bytes on wire, 260 bytes captured) *
*Ethernet II, Src: 00:11:0c:20:4e:32, Dst: 00:11:0c:29:4e:32, Len: 40 (40 bytes) *
*Frame 371 (260 bytes on wire, 260 bytes captured) *
*Ethernet II, Src: 00:11:0c:20:4e:32, Dst: 00:11:0c:29:4e:32, Len: 40 (40 bytes) *
*Frame 371 (260 bytes on wire, 260 bytes captured) *
*Ethernet II, Src: 00:11:0c:20:4e:32, Dst: 00:11:0c:29:4e:32, Len: 40 (40 bytes) *
*Frame 371 (260 bytes on wire, 260 bytes captured) *
*Ethernet II, Src: 00:11:0c:20:4e:32, Dst: 00:11:0c:29:4e:32, Len: 40 (40 bytes) *
*Frame 371 (260 bytes on wire, 260 bytes captured) *
*Ethernet II, Src: 00:11:0c:20:4e:32, Dst: 00:11:0c:29:4e:32, Len: 40 (40 bytes) *

Figure 3.23: Integrated operation of HIP and SIP - Mobility

58
Chapter 4

Conclusions

4.1 Objectives Review

When starting the project, we proposed to fully test the HIP protocol by establishing a testbed that would permit experimenting with all of the available HIP implementations. We also proposed to elaborate a HIP manual, that would be a result of all the gathered knowledge, theoretical and empirical.

For the second part of the project we proposed the implementation of a SIP-based application with which we could test mobility procedures at the application level. For this part of the work, we tried to develop a solution, based on the REFER extension to SIP, that would permit the transference of a data stream between two machines, i.e., application mobility.

4.2 Results

At the end of the project we've experimented with all of the available implementations. Due to release dates of each implementation, and the deadline of this project, some capabilities we're not experimented. Namely, DNS integration that is currently supported by OpenHIP and HIPL. It would be interesting to add this capability to our test-bed that would allow the resolution of SIP URIs (section 2.3.3).

As expected, the test-bed permitted a empirical learning experience of HIP. This empirical approach was supported by theoretical investigation of the protocol. The result is the elaboration of a HIP manual, that is expected to act as future reference. However, it must be kept in mind that HIP is still in a development fase and despite most of the protocol being already defined, some alterations are expected, particularly those related to scalability and deployment.

For the second part of the work, a SIP-based application was developed. This application acts based on the client-server paradigm. Its operation consists on the transmission of a media stream from the Server to the Client after a SIP signaling procedure occurs. This application add as one of the objectives, to experiment with the REFER extension to SIP. We proposed to make some alterations to the REFER extension so that the transference of an ongoing media stream between two client machines would be possible. As said in the SIP Application section in the Work Description chapter (section 3.3.4), this objective wasn's fully achieved, but the transference procedure was successfully emulated in the Server by applying some small modifications to the
Client application running on one of the machines. The application was also able to receive a media stream, after sending the request parameters using SIP and ending that same session. A GUI was developed that permitted a user-friendly interaction with the application.

The development of this application was a good opportunity to develop programming skills in a Java platform. This application was developed using the good practices and procedures of Software Engineering which also constituted a good learning experience.

4.3 Future Work

The future work related to this project is closely affected by the future development of HIP and its implementations. The continuous integration of more capabilities in the testbed would permit a development of the knowledge acquired until now. At this moment, DNS integration would be the next step and HIPL has a new version with Hi3 [2] support.

From the application point of view, the next steps would be the correct use of the REFER extension, experimenting different ways of implementation, permitting performance and security evaluations. Also, the integration of HIP with DNS would permit the use of SIP URIs in the application. Minor changes would be required to the application so it could support SIP URIs. After a testing implementation, another implementation with optimized coding would permit an quantitative evaluation of the hole testbed and the combined mobility scenario.
Appendix A

Host Identity Protocol Manual

A.1 Introduction

The current Internet uses two global namespaces: domain names and IP addresses. The first namespace - domain names - has a single use. Usually simply referred as names, they're symbolic identifiers for sets of numeric IP addresses, chosen for their mnemonic properties: humans need to interact with them. IP addresses form the Internet's second global namespace. They have two uses: First, they are topological locators for network attachment points, addressing a specific location in the network topology. Their second use is as identifiers for the network interfaces - and thus nodes - that attach to the addressed locations. In this role as identifiers, IP addresses lose their topological meaning and become simple names. Routing and other network-layer mechanisms use the locator aspects of IP addresses. Transport-layer protocols and mechanisms typically use IP addresses in their role as names for communication endpoints. [23]

<table>
<thead>
<tr>
<th>TCP / UDP / ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIP</td>
</tr>
<tr>
<td>IP</td>
</tr>
</tbody>
</table>

Figure A.1: HIP in the Networking Stack - Layer 2.5

The Host Identity Protocol (HIP) is a multi-addressing and mobility solution for the IPv4 and IPv6 Internet with always-on security policy. HIP adds a new Host Identity (HI) layer between the transport and network layers [18]. The new layer maps the host identifiers to network addresses and vice-versa. This achieves the main architectural goal of HIP: the separation of identifiers from locators, which is the main driving force behind mobility and multihoming. The new layer also provides host-to-host authentication, identifying the hosts using their public keys or the hash values of public keys.
The new architecture will provide intrinsic security and support for IP-layer mobility and multi-homing [17].

Features of the HIP protocol, such as authentication and encryption, mobility and multi-homing, and protection from Denial of Service attacks, are useful for all applications. In addition, the new cryptographic namespace introduced by HIP may be used to create new novel applications and improve existing ones.

HIP is expected to provide solutions to some of the Internet’s shortcomings and current challenges, like Host Mobility, Host Multihoming, Site Multihoming and Network Level Security. Due to the format of it’s identifiers and the abstraction from the IP layer (much like DNS), HIP could also smooth the transition to IPv6 networking.[20]. In addition, the new cryptographic namespace introduced by HIP may be used to create new novel applications and improve existing ones.

A.2 Terminology

In this section we'll introduce some of the components that are part of the HIP architecture.

A.2.1 Host Identity Namespace

A name in the Host Identity Namespace - a Host Identifier (HI) - represents a statistically globally unique name for naming any system with an IP stack.

This identity is normally associated with, but not limited to, an IP stack. A system can have multiple identities, some 'well known', some unpublished or 'anonymous'. A system may self-assert its own identity, or may use a third-party authenticator like DNS Security (DNSSEC) [1], Pretty Good Privacy (PGP) [5], or X.509 [9] to 'notarize' the identity assertion. It is expected that the Host Identifiers will initially be authenticated with DNSSEC and that all implementations will support DNSSEC as a minimal baseline.

In theory, any name that can claim to be 'statistically globally unique' may serve as a Host Identifier. However a public key of a 'public key pair' was chosen for HIP. A public-key-based HI can authenticate the HIP packets and protect them from man-in-the-middle attacks. Since authenticated datagrams are mandatory to provide much of HIP's DoS protection, the Diffie-Hellman exchange in HIP has to be authenticated. Thus, only public key HI and authenticated HIP messages are supported in practice.

A.2.2 Host Identifier - HI

The Host Identifier is the public part of a public/private key pair that univocally identifies a host in the Host Identity Namespace.

The Host Identity adds two main features to Internet protocols: The first is a decoupling of the internetworking and transport layers. This decoupling will allow for independent evolution of the two layers. In addition, it can provide end-to-end services over multiple internetworking realms. The second feature is host authentication. Because the Host Identifier is a public key, this key can be used for authentication in security protocols like IPsec. The possession of the private part of the public/private key pair defines the Identity itself. If the private key is possessed by more than one node, the Identity can be considered to be a distributed one.
Architecturally, any other Internet naming convention might form a usable base for Host Identifiers. However, non-cryptographic names should only be used in situations of high trust/low risk, that is, any place where host authentication is not needed (no risk of host spoofing and no use of IPsec). However, at least for interconnected networks spanning several operational domains, the set of environments where the risk of host spoofing allowed by non-cryptographic Host Identifiers is acceptable is non-existent. Hence, the current HIP documents do not specify how to use any other types of Host Identifiers but public keys.

The actual Host Identifiers are never directly used in any Internet protocols. The corresponding Host Identifiers (public keys) may be stored in various DNS or Lightweight Directory Access Protocol (LDAP), and passed in the HIP base exchange. A Host Identity Tag (HIT) is used in other protocols to represent the Host Identity. Another representation of the Host Identities, the Local Scope Identifier (LSI), can also be used in protocols and APIs.

A.2.3 Host Identity Tag- HIT

A Host Identity Tag is a 128-bit representation for a Host Identity.

Its format, similar to IPv6 (e.g. 1100::/8) but without any structure, is created by taking a cryptographic hash over the corresponding Host Identifier. There are two advantages of using a hash over using the Host Identifier in protocols. First, its fixed length makes for easier protocol coding and also better manages the packet size cost of this technology. Second, it presents the identity in a consistent format to the protocol independent of the cryptographic algorithms used.

In the HIP packets, the HITs identify the sender and recipient of a packet. Consequently, a HIT should be unique in the whole IP universe as long as it is being used. In the extremely rare case of a single HIT mapping to more than one Host Identity, the Host Identifiers (public keys) will make the final difference. If there is more than one public key for a given node, the HIT acts as a hint for the correct public key to use.

Generating a HIT from a HI

The HIT must be generated according to the KHI generation method described in [19] using a context ID value of 0xF0EF F02F BFF4 3D0F E793 0C3C 6E61 74EA, and an input encoding the Host Identity field present in a HIP payload packet. Please refer to section A.4.4 for a in-depth description of the HIP packets. For Identities that are either RSA[12] or DSA public keys, this input consists of the public key encoding as specified in the corresponding DNSSEC document, taking the algorithm specific portion of the RDATA part of the KEY RR. There is currently only two defined public key algorithms: RSA and DSA. Hence, either of the following applies:

- The RSA public key is encoded as defined in RFC3110 Section 2, taking the exponent length e_len, exponent e and modulus n fields concatenated. The length n_len of modulus n can be determined from the total HI Length and the preceding HI fields including the exponent e. Thus, the data to be hashed has the same length as the HI. The fields must be encoded in network byte order, as defined in RFC3110 [15].

- The DSA public key is encoded as defined in RFC2536 [13] Section 2, taking the fields T, Q, P, G, and Y, concatenated. Thus, the data to be hashed is $1 + 20 + 3 \times 64 + 3 \times 8 \times T$ octets long, where T is the size parameter as defined in RFC2536
[13]. The size parameter T, affecting the field lengths, must be selected as the minimum value that is long enough to accommodate P, G, and Y. The fields must be encoded in network byte order, as defined in RFC2536 [13].

As an example, the following pseudo-codes illustrate the process to generate a public key encoding from a HI for both RSA and DSA.

The symbol := denotes assignment; the symbol + = denotes appending. The pseudo-function encode_in_network_byte_order takes two parameters, an integer bignum and a length in bytes, and returns the integer encoded into a byte string of the given length.

```c
switch ( HI.algorithm )
{
  case RSA:
    buffer := encode_in_network_byte_order ( HI.RSA.e_len,  
       ( HI.RSA.e_len > 255 ) ? 3 : 1 )
    buffer += encode_in_network_byte_order ( HI.RSA.e, HI.RSA.e_len )
    buffer += encode_in_network_byte_order ( HI.RSA.n, HI.RSA.n_len )
    break;
  case DSA:
    buffer := encode_in_network_byte_order ( HI.DSA.T , 1 )
    buffer += encode_in_network_byte_order ( HI.DSA.Q , 20 )
    buffer += encode_in_network_byte_order ( HI.DSA.P , 64 + 8 * HI.DSA.T )
    buffer += encode_in_network_byte_order ( HI.DSA.G , 64 + 8 * HI.DSA.T )
    buffer += encode_in_network_byte_order ( HI.DSA.Y , 64 + 8 * HI.DSA.T )
    break;
}
```

A.2.4 Local Scope Identifier- LSI

A Local Scope Identifier is a 32-bit localized representation for a Host Identity. In many environments, 128 bits is still considered large. For example, currently used IPv4-based applications are constrained with 32-bit address fields. Another problem is that the cohabitation of IPv6 and HIP might require some applications to differentiate an IPv6 address from a HIT. So, the purpose of an LSI is to facilitate using Host Identities in existing protocols and APIs. There are two types of such LSIs: 32-bit IPv4-compatible ones and 128-bit IPv6-compatible ones. The LSI provides a compression of the HIT with only a local scope so that it can be carried efficiently in any application level packet and used in API calls. The LSIs do not have the same properties as HITs (i.e., they are not self-certifying nor are they as unlikely to collide - hence their local scope), and consequently they must be used more carefully.

By default, the LSI is obtained using concatenating the binary value of 0000 0001 with the 24 LSB of the HIT with This produces a LSI with the format 1.xxx.xxx.xxx. The biggest advantages of LSI over HIT are it's size and ability to work with legacy APIs. Its major disadvantage is its localized application.
A.3 IPSec Background

Here we present some IPSec concepts necessary for the full comprehension of the HIP operation discussed ahead.

A.3.1 Diffie-Hellman

The Diffie-Hellman (D-H) key exchange is a cryptographic protocol which allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.

Operation

Two hosts independently and randomly generate values much like a public/private key pair. Each sends its public value to the other (using authentication to close out the man-in-the-middle). Each then combines the public key just received with the private key just generated, using the Diffie-Hellman combination algorithm. The resulting value is the same on both sides, and therefore can be used for fast symmetric encryption by both parties. Theoretically, no one else come up with the same value from the two public keys passed through the net, since the final value also depends on the private values, which remain secret.

A.3.2 IPSec Security Associations - SAs

A Security Association (SA) is a simplex "connection" that affords security services to the traffic carried by it. Security services are afforded to an SA by the use of AH, or ESP, but not both[15]. If both AH and ESP protection is applied to a traffic stream, then two (or more) SAs are created to afford protection to the traffic stream. To secure typical, bi-directional communication between two hosts, or between two security gateways, two Security Associations (one in each direction) are required.

Security Parameters Index - SPI

The SA is a concept, whereas the SPI is more concrete. The SPI is a number that uniquely identifies an SA. The SPI, together with the SA concept, makes keeping track of keys and protocols easy and automatic. Specifically, the SPI is an arbitrary 32-bit number your system picks to represent that SA whenever someone negotiates an SA with you. The SPI can not be encrypted in the packet because it is needed in the latter decryption of the packet. Upon negotiation of an SA, the recipient node assigns an SPI it isn't already using, and preferably, one it hasn't used in a while. It then communicates this SPI to the node with which it negotiated the SA. From then until that SA expires, whenever that node wishes to communicate with yours using that SA, it uses that SPI to specify it.

When receiving a packet, the node looks at the SPI to determine which SA it needs to use. Then it authenticates and/or decrypts the packet according to the rules of that SA, using the agreed-upon keys and algorithms to verify that the data really did come from the node it claims, that the data has not been modified, and that no one between those nodes has read the data.
A.4 HIP Operation

In this section we'll analyze the operation of the HIP, covering several pertinent scenarios.

A.4.1 Base Exchange

The HIP base exchange serves to manage the establishment of state between an Initiator and a Responder.

By definition, the system initiating a HIP exchange is the Initiator, and the peer is the Responder. This distinction is forgotten once the base exchange completes, and either party can become the Initiator in future communications. The Base Exchange is used to establish a pair of IPsec security associations (SA) (see section A.3.2 for IPsec details) between two HIP systems, also called HIP hosts. After the the Base Exchange is finished, the data is encapsulated using, by default, IPsec's ESP. Thus, we have two different network stacks: one for the signaling plane with HIP between layers 2 and 3 and another for the data plane with an ESP layer on the same position.

Figure A.2: On the left: Networking Stack - Control Plane. On the right: Network Stack - Data Plane (Using ESP)

The first packet, I1, initiates the exchange, and the last three packets, R1, I2, and R2, constitute a standard authenticated Diffie-Hellman (see section A.3.1) key exchange for session key generation. During the Diffie-Hellman key exchange, a piece of keying material is generated. The HIP association keys are drawn from this keying material. If other cryptographic keys are needed, e.g., to be used with ESP, they are expected to be drawn from the same keying material.

II - The HIP Initiator Packet

The first packet sent by the Initiator - II has as only function to trigger the Base Exchange with the Responder. This packet must carry the Initiator's HIT in the header. The Responder's HIT is also usually present in the header, defining the destination of the packet, but is not mandatory. In this case the Base Exchange is initiated in opportunistic mode.

The HIP header values for the II packet:
Header:
Packet Type = 1
SRC HIT = Initiator’s HIT
DST HIT = Responder’s HIT, or NULL

IP (HIP())

The H1 packet contains only the fixed HIP header. Valid control bits: none.

The Initiator gets the Responder’s HIT either from a DNS (see the Integration with DNS in section A.4.6) lookup of the Responder’s Fully Qualified Domain Name (FQDN) from some other repository, or from a local table. If the Initiator does not know the Responder’s HIT, it may attempt opportunistic mode by using NULL (all zeros) as the Responder’s HIT. If the Initiator sends a NULL as the Responder’s HIT, it must be able to handle all both of the currently supported cryptic algorithms - RSA and DSA. Since this packet is so easy to spoof even if it were signed, no attempt is made to add to its generation or processing cost. Implementations must be able to handle a storm of received H1 packets, discarding those with common content that arrive with a small time interval between them.

R1 - The HIP Responder Packet

The Responder will answer the Initiator’s H1 message with a R1 packet.

However, even before the Responder receives the H1, it precomputes a partial R1 message. The precomputed R1 includes the HIT-R, the Responder’s Diffie-Hellman key, the Responder host identity HI-R (i.e., a public key), the proposed cryptographic algorithms for the rest of the base exchange (HIP transforms), the proposed IPsec algorithms (ESP transforms), and an Echo_Request field. The Echo_Request contains data that the Initiator returns unmodified in the following message I2. It is important that the responder sends R1 without creating any protocol state. The Echo_Request can be used to store some data in a stateless way. The responder signs the message. The HIT-I and the Puzzle field are left empty at this point. These two fields are populated after receiving an I1 and they are not protected by the signature. The Puzzle parameter in R1 contains a cryptographic puzzle, which the Initiator is required to solve before sending the following packet I2. The idea is that the Initiator is forced to perform a moderately expensive brute-force computation before the Responder commits its computational resources to the protocol or creates a protocol state. This procedure prevents
against Denial-of-Service attacks.

The HIP header values for the R1 packet:

Header:
Packet Type = 2
SRC HIT = Responder's HIT
DST HIT = Initiator's HIT

IP ( HIP ( [ R1_COUNTER, ]
   PUZZLE,
   DIFFIE_HELLMAN,
   HIP_TRANSFORM,
   HOST_ID,
   [ ECHO_REQUEST, ]
   HIP_SIGNATURE_2 )
   [, ECHO_REQUEST ]
)

Valid control bits: A
If the Responder HI is an anonymous one, the A control must be set.

The Initiator HIT must match the one received in I1. If the Responder has multiple HIs, the Responder HIT used must match Initiator's request. If the Initiator used opportunistic mode, the Responder may select freely among its own HIs.

- **R1_COUNTER**
The R1 generation counter is used to determine the currently valid generation of puzzles. The value is increased periodically, and it is recommended that it is increased at least as often as solutions to old puzzles are no longer accepted.

- **PUZZLE**
The Puzzle contains a random #I and the difficulty K. The difficulty K is the number of bits that the Initiator must get zero in the puzzle. The random #I is not covered by the signature and must be zeroed during the signature calculation, allowing the sender to select and set the #I into a pre-computed R1 just prior sending it to the peer. The puzzle has three components: the puzzle nonce I, the difficulty level K, and the solution J. A puzzle solution is verified by:

1. Concatenation of I, with host identity tags HIT-I and HIT-R, and the solution J.
2. Computing the SHA-1 hash of the concatenation.
3. Checking that the K low-order bits of the hash are all zeros.

The Initiator must do a brute-force search for the value of J, which takes $2^K$ trials. The Responder, on the other hand, can verify the solution by computing a single hash.

- **DIFFIE_HELLMAN**
The Diffie-Hellman value is ephemeral, but can be reused over a number of connections. In fact, as a defense against I1 storms, an implementation may use the same Diffie-Hellman value for a period of time, for example, 15 minutes. By using a small number of different puzzles for a given Diffie-Hellman value, the
R1 packets can be pre-computed and delivered as quickly as R1 packets arrive. A scavenger process should clean up unused DHs and puzzles.

- **HIP_TRANSFORM**
  The HIP_TRANSFORM field contains the encryption and integrity algorithms supported by the Responder to protect the HI exchange, in the order of preference. All implementations must support the AES with HMAC-SHA-1-96.

- **ECHO_REQUEST**
  The ECHO_REQUEST field contains data that the sender wants to receive unmodified in the corresponding response packet in the ECHO_RESPONSE parameter. The ECHO_REQUEST can be either covered by the signature, or it can be left out from it. In the first case, the ECHO_REQUEST gets Type number 897 and in the latter case 63661.

  The signature is calculated over the whole HIP envelope, after setting the Initiator HIT, header checksum as well as the Opaque field and the Random #1 in the PUZZLE parameter temporarily to zero, and excluding any parameters that follow the signature. This allows the Responder to use precomputed R1s. The Initiator should validate this signature. It should check that the Responder HI received matches with the one expected, if any.

**I2 - The Second HIP Initiator Packet**

On receiving R1, the initiator checks that it has sent a corresponding I1 and verifies the signature using the public key HI-R. If the signature is ok, it solves the puzzle and creates the message I2. I2 includes the puzzle and its solution, the Initiator's Diffie-Hellman key, the HIP and ESP transforms proposed by the Initiator, a security parameter index (SPI) for the Responder-to-Initiator IPsec SA, the Initiator public key (HI-I) encrypted using the new session key, and the Echo_Response. A signature covers the entire message. Key material for the session keys is computed as a SHA-1[6] hash of the Diffie-Hellman shared secret Kij.

The HIP header values for the I2 packet:

- **Header:**
  
  `Type = 3`
  `SRC HIT = Initiator's HIT`
  `DST HIT = Responder's HIT`

- **IP ( HIP ( [R1_COUNTER],)
  SOLUTION,
  DIFFIE_HELLMAN,
  HIP_TRANSFORM,
  ENCRYPTED { HOST_ID } or HOST_ID,
  [ ECHO_RESPONSE ],
  HMAC,
  HIP_SIGNATURE
  [, ECHO_RESPONSE] )

Valid control bits: A
The HITs used must match the ones used previously.
If the Initiator HI is an anonymous one, the A control must be set.
• **R1_COUNTER**
  The Initiator may include an unmodified copy of the R1_COUNTER parameter received in the corresponding R1 packet into the I2 packet.

• **SOLUTION**
  The Solution contains the random #I from R1 and the computed #J. The low order K bits of the PHASH(I ... | J) must be zero.

• **DIFFIE_HELLMAN**
  The Diffie-Hellman value is ephemeral. If precomputed, a scavenger process should clean up unused DHs.

• **HIP_TRANSFORM**
  The HIP_TRANSFORM contains the single encryption and integrity transform selected by the Initiator, that will be used to protect the HI exchange. The chosen transform must correspond to one offered by the Responder in the R1. All implementations must support the AES transform.

• **ENCRYPTED { HOST_ID } or HOST_ID**
  The Initiator’s HI may be encrypted using the HIP_TRANSFORM encryption algorithm. The keying material is derived from the Diffie-Hellman exchange.

• **ECHO_RESPONSE**
  The ECHO_RESPONSE contains the unmodified Opaque data copied from the corresponding ECHO_REQUEST parameter. The ECHO_RESPONSE can be either covered by the HMAC and SIGNATURE or not covered. In the former case, the ECHO_RESPONSE gets Type number 961, in the latter it is 63425.

• **HMAC**
  The HMAC is calculated over whole HIP envelope, excluding any parameters after the HMAC. The Responder must validate the HMAC.

• **HIP_SIGNATURE**
  The signature is calculated over whole HIP envelope, excluding any parameters after the HIP_SIGNATURE. The Responder must validate this signature. It may use either the HI in the packet or the HI acquired by some other means.

**R2 - The Second HIP Responder Packet**

The HIP header values for the R2 packet:

Header:
- **Packet Type** = 4
- **SRC HIT** = Responder’s HIT
- **DST HIT** = Initiator’s HIT

\[
\text{IP ( HIP ( HMAC_2, HIP\_SIGNATURE ) )}
\]

Valid control bits: none
• HMAC_2
  The HMAC_2 is calculated over whole HIP envelope, with Responder's HOST_ID parameter concatenated with the HIP envelope. The HOST_ID parameter is removed after the HMAC calculation.

• HIP_SIGNATURE
  The HIP_SIGNATURE is calculated over whole HIP envelope.

The Initiator MUST validate both the HMAC and the signature.

A.4.2 Simplified HIP State Diagram

The following diagram in fig.A.4 shows the major state transitions. Transitions based on received packets implicitly assume that the packets are successfully authenticated or processed.

A.4.3 The Rendevouz Server Mechanism

HIP introduces a new mechanism based on the paradigm of the indirection architectures [25].

This paradigm assumes the existence of a physical or a logical indirection point interposed between the sender and the receiver(s) that relays the traffic between them. By communicating through the indirection point rather than directly to the end-host, a sender can abstract away the location and the number of receivers. For instance, mobile IP assumes a home agent that hides the end-host mobility, while IP multicast assumes a logical indirection point (address) that hides the number of receivers and their locations. In the case of HIP, this indirection point is called Rendevouz Server.

Terminology

Rendevouz Service
A HIP service provided by a rendezvous server to its rendezvous clients. The rendezvous server offers to relay some of the arriving base exchange packets between the initiator and responder.

Rendevouz Server - RVS
A HIP registrar providing rendezvous service.

Rendevouz Client
A HIP requester that has registered for rendezvous service at a rendezvous server.

Rendevouz Registration
A HIP registration for rendezvous service, established between a rendezvous server and a rendezvous client.
The Rendezvous Server Operation

A HIP node may want to be reachable to future correspondent peers that are unaware of its location change. The HIP architecture introduces rendezvous servers with whom a HIP node may register its host identity tags (HITs) and current IP addresses. An RVS relays HIP packets arriving for these HITs to the node's registered IP addresses. When a HIP node has registered with an RVS, it should record the IP address of its RVS in its DNS record, using the HIPRVS DNS record type defined in [I-D.ietf-hip-dns].

Figure A.5 shows a HIP base exchange involving a rendezvous server. It is assumed that HIP node Receiver R previously registered its HITs and current IP addresses with the RVS, using the HIP registration protocol [16]. When the Initiator I tries to establish contact with the responder R, it must send the II of the base exchange either to one of R's IP addresses (if known via DNS or other means) or to one of R's rendezvous servers instead. Here, I obtains the IP address of R's rendezvous server from R's DNS record and then sends the II packet of the HIP base exchange to RVS.

The RVS, noticing that the HIT contained in the arriving II packet is not one of its own, must check its current registrations to determine if it needs to relay the packets. Here, it determines that the HIT belongs to R and then relays the III packet to the registered IP address. R then completes the base exchange without further assistance from RVS by sending an RI directly to the I's IP address, as obtained from the II packet. In this specification the client of the RVS is always the responder. However, there might be reasons to allow a client to initiate a base exchange through its own RVS, like NAT and firewall traversal. The current specification does not address such scenarios.

Rendezvous Client Registration

Before a rendezvous server starts to relay HIP packets to a rendezvous client, the rendezvous client needs to register with it to receive rendezvous service by using the HIP registration extension [I-D.ietf-hip-registration] as illustrated in the following schema:

Relaying the Base Exchange

If a HIP node and one of its rendezvous servers have a rendezvous registration, the rendezvous servers relay inbound II packets that contain one of the client's HITs by rewriting the IP header. They replace the destination IP address of the II packet with one of the IP addresses of the owner of the HIT, i.e., the rendezvous client. They must also recompute the IP checksum accordingly.

Because of egress filtering on the path from the RVS to the client [RFC2827][RFC3013], a HIP rendezvous server should replace the source IP address, i.e., the IP address of I, with one of its own IP addresses. The replacement IP address should be chosen according to [RFC1122] and, when IPv6 is used, to [RFC3484]. Because this replacement conceals the initiator's IP address, the RVS must append a FROM parameter containing the original source IP address of the packet. This FROM parameter must be integrity protected by an RVS_HMAC keyed with the corresponding rendezvous registration integrity key [I-D.ietf-hip-registration].

This modification of HIP packets at a rendezvous server can be problematic because the HIP protocol uses integrity checks. Because the II does not include HMAC or SIGNATURE parameters, these two end-to-end integrity checks are unaffected by the operation of rendezvous servers.
The RVS should verify the checksum field of an II packet before doing any modifications. After modification, it must recompute the checksum field using the updated HIP header, which possibly included new FROM and RVS_HMAC parameters, and a pseudo-header containing the updated source and destination IP addresses. This enables the responder to validate the checksum of the II packet "as is", without having to parse any FROM parameters.

Processing Outgoing II Packets

An initiator should not send an opportunistic II with a NULL destination HIT to an IP address which is known to be a rendezvous server address, unless it wants to establish a HIP association with the rendezvous server itself and does not know its HIT.

As referenced in the previous section, an RVS rewrites the source IP address of an II packet due to egress filtering, it must add a FROM parameter to the II that contains the initiator's source IP address. This FROM parameter must be protected by an RVS_HMAC keyed with the integrity key established at rendezvous registration.

Processing Incoming II Packets

When a rendezvous server receives an II whose destination HIT is not its own, it consults its registration database to find a registration for the rendezvous service established by the HIT owner. If it finds an appropriate registration, it relays the packet to the registered IP address. If it does not find an appropriate registration, it drops the packet.

A rendezvous server should interpret any incoming opportunistic II (i.e., an II with a NULL destination HIT) as an II addressed to itself and should not attempt to relay it to one of its clients.

When a rendezvous client receives an II, it must validate any present RVS_HMAC parameter. If the RVS_HMAC cannot be verified, the packet should be dropped. If the RVS_HMAC cannot be verified and a FROM parameter is present, the packet must be dropped.

A rendezvous client acting as responder should drop opportunistic IIIs that include a FROM parameter, because this indicates that the II has been relayed.

Processing Outgoing R1 Packets

When a responder replies to an II relayed via an RVS, it must append to the regular R1 header a VIA_RVS parameter containing the IP addresses of the traversed RVS's.

Processing Incoming R1 Packets

The HIP base specification [**I-D.ietf-hip-base] mandates that a system receiving an R1 must first check to see if it has sent an II to the originator of the R1 (i.e., it is in state II-SENT). When the R1 is replying to a relayed II, this check should be based on HITs only. In case the IP addresses are also checked, then the source IP address must be checked against the IP address included in the VIA_RVS parameter.

A.4.4 Description of the HIP packets

All HIP packets start with a fixed header.
- **Next Header**: The HIP header is logically an IPv6 extension header. However, there is no specification for the Next Header values other than decimal 59, IPPROTO_NONE, the IPv6 no next header value. Future specifications may do so.

- **Header Length**: The Header Length field contains the length of the HIP Header and HIP parameters in 8 bytes units, excluding the first 8 bytes. Since all HIP headers must contain the sender’s and receiver’s HIT fields, the minimum value for this field is 4.

- **Packet Type**: The Packet Type indicates the HIP packet type. If a HIP host receives a HIP packet that contains an unknown packet type, it must drop the packet.

- **VER.**: The HIP Version is four bits. The current version is 1. The version number is expected to be incremented only if there are incompatible changes to the protocol. Most extensions can be handled by defining new packet types, new parameter types, or new controls.

- **RES.**: The following three bits are reserved for future use. They must be zero when sent, and they should be ignored when handling a received packet.

- **Checksum**: Since the checksum covers the source and destination addresses in the IP header, it must be recomputed on HIP-aware NAT devices.

- **Controls**: The HIP Controls section conveys information about the structure of the packet and capabilities of the host. The following fields have been defined:
A - Anonymous: If this is set, the sender’s HI in this packet is anonymous, i.e., one not listed in a directory. Anonymous HIs should not be stored. This control is set in packets R1 and/or I2. The peer receiving an anonymous HI may choose to refuse it. The rest of the fields are reserved for future use and must be set to zero on sent packets and ignored on received packets.

- **HIP Parameters:** The HIP Parameters are used to carry the public key associated with the sender’s HIT, together with related security and other information. They consist of ordered parameters, encoded in TLV (Type/Length/Value) format. The following parameter types are currently defined.

<table>
<thead>
<tr>
<th>TLV</th>
<th>Type</th>
<th>Length</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1_COUNTER</td>
<td>128</td>
<td>12</td>
<td>System Boot Counter</td>
</tr>
<tr>
<td>PUZZLE</td>
<td>257</td>
<td>12</td>
<td>K and Random #I</td>
</tr>
<tr>
<td>SOLUTION</td>
<td>321</td>
<td>20</td>
<td>K, Random #I and puzzle solution J</td>
</tr>
<tr>
<td>SEQ</td>
<td>385</td>
<td>4</td>
<td>Update packet ID number</td>
</tr>
<tr>
<td>ACK</td>
<td>449</td>
<td>variable</td>
<td>Update packet ID number</td>
</tr>
<tr>
<td>DIFFIE_HELLMAN</td>
<td>513</td>
<td>variable</td>
<td>public key</td>
</tr>
<tr>
<td>HIP_TRANSFORM</td>
<td>577</td>
<td>variable</td>
<td>HIP Encryption and Integrity Transform</td>
</tr>
<tr>
<td>ENCRYPTED</td>
<td>641</td>
<td>variable</td>
<td>Encrypted part of I2 packet</td>
</tr>
<tr>
<td>HOST_ID</td>
<td>705</td>
<td>variable</td>
<td>Host Identity with Fully Qualified Domain Name or NAI</td>
</tr>
<tr>
<td>CERT</td>
<td>768</td>
<td>variable</td>
<td>I Certificate; used to transfer certificates. Usage defined in a separate document.</td>
</tr>
<tr>
<td>NOTIFY</td>
<td>832</td>
<td>variable</td>
<td>Informational data</td>
</tr>
<tr>
<td>ECHO_REQUEST</td>
<td>897</td>
<td>variable</td>
<td>Opaque data to be echoed back; under signature</td>
</tr>
<tr>
<td>ECHO_RESPONSE</td>
<td>961</td>
<td>variable</td>
<td>Opaque data echoed back; under signature</td>
</tr>
<tr>
<td>HMAC</td>
<td>61505</td>
<td>variable</td>
<td>HMAC based message authentication code, with key material from HIP_TRANSFORM</td>
</tr>
<tr>
<td>HMAC_2</td>
<td>61569</td>
<td>variable</td>
<td>HMAC based message authentication code, with key material from HIP_TRANSFORM</td>
</tr>
<tr>
<td>HIP_SIGNATURE_2</td>
<td>61633</td>
<td>variable</td>
<td>Signature of the R1 packet</td>
</tr>
<tr>
<td>HIP_SIGNATURE</td>
<td>61697</td>
<td>variable</td>
<td>Signature of the packet</td>
</tr>
<tr>
<td>ECHO_REQUEST</td>
<td>63661</td>
<td>variable</td>
<td>Opaque data to be echoed back; after signature</td>
</tr>
<tr>
<td>ECHO_RESPONSE</td>
<td>63425</td>
<td>variable</td>
<td>Opaque data echoed back; after signature</td>
</tr>
</tbody>
</table>

Further information about all the HIP Parameters can be found in [**I-D.ietf-hip-base](https://tools.ietf.org/html/draft-ietf-hip-base-00).
The HIT fields are always 128 bits (16 bytes) long.

A.4.5 Mobility

Mobility in HIP can be split into two contexts:

- Session Mobility, that addresses the mobility scenario during a communication session.
- Global Mobility, which refers to pre-session mobility. In this scenario, mobility support is provided by the use of a RVS, as seen on section A.4.3.

The UPDATE Packet

Support for the UPDATE packet is mandatory.

The HIP header values for the UPDATE packet:

Header:
- Packet Type = 16
- SRC HIT = Sender’s HIT
- DST HIT = Recipient’s HIT

IP ( HIP [ SEQ, ACK, ] HMAC, HIP_SIGNATURE )

Valid control bits: None

The UPDATE packet contains mandatory HMAC and HIP_SIGNATURE parameters, and other optional parameters. It also contains zero or one SEQ parameter. The presence of a SEQ parameter indicates that the receiver must acknowledge the UPDATE. An UPDATE that does not contain a SEQ parameter is simply an ACK of a previous UPDATE and itself must not be acked.

Session Mobility

When a host moves to another address, it notifies its peer of the new address by sending a HIP UPDATE packet containing a LOCATOR parameter. This UPDATE packet is acknowledged by the peer, and is protected by retransmission. The peer can authenticate the contents of the UPDATE packet based on the signature and keyed hash of the packet.

When using ESP Transport Format [14], the host may at the same time decide to rekey its security association and possibly generate a new Diffie-Hellman key (see section A.3 for IPSec details). All of these actions are triggered by including additional parameters in the UPDATE packet, as defined in the base protocol specification [**I-D.ietf-hip-dns] and ESP extension [I-D.ietf-hip-esp]

- Mobility with single SA pair (no rekeying)
  A mobile host must sometimes change an IP address bound to an interface. The change of an IP address might be needed due to a change in the advertised IPv6 prefixes on the link, a reconnected PPP link, a new DHCP lease, or an actual movement to another subnet. In order to maintain its communication context, the host must inform its peers about the new IP address.
For the simplest mobility example we consider the case in which the mobile host has only one interface, IP address, a single pair of SAs (one inbound, one outbound), and no rekeying occurs on the SAs. We also assume that the new IP addresses are within the same address family (IPv4 or IPv6) as the first address.

- **UPDATE 1**
  The mobile host is disconnected from the peer host for a brief period of time while it switches from one IP address to another. Upon obtaining a new IP address, the mobile host sends a LOCATOR parameter to the peer host in an UPDATE message. The UPDATE message also contains an ESP_INFO parameter with the "Old SPI" and "New SPI" parameters both set to the value of the pre-existing incoming SPI. This ESP_INFO does not trigger a rekeying event but is instead included for possible parameter-inspecting middleboxes on the path. The LOCATOR parameter contains the new IP address (Locator Type of "I", defined below) and a locator lifetime. The mobile host waits for this UPDATE to be acknowledged, and retransmits if necessary, as specified in the base specification [**2].

- **UPDATE 2**
  The peer host receives the UPDATE, validates it, and updates any local bindings between the HIP association and the mobile host's destination address. The peer host must perform an address verification by placing a nonce in the ECHO_REQUEST parameter of the UPDATE message sent back to the mobile host. It also includes an ESP_INFO parameter with the "Old SPI" and "New SPI" parameters both set to the value of the pre-existing incoming SPI, and sends this UPDATE (with piggybacked acknowledgment) to the mobile host at its new address. The peer may use the new address immediately, but it must limit the amount of data it sends to the address until address verification completes.

- **UPDATE 3**
  The mobile host completes the readdress by processing the UPDATE ACK and echoing the nonce in an ECHO_RESPONSE. Once the peer host receives this ECHO_RESPONSE, it considers the new address to be verified and can put it into full use.

While the peer host is verifying the new address, the new address is marked as UNVERIFIED in the interim, and the old address is DEPRECATED. Once the peer host has received a correct reply to its UPDATE challenge, it marks the new address as ACTIVE and removes the old address.

- **Mobility with single SA pair (mobile-initiated rekey)**
  The mobile host may decide to rekey the SAs at the same time that it is notifying the peer of the new address. In this case, the above procedure described in Figure 3 is slightly modified. The UPDATE message sent from the mobile host includes an ESP_INFO with the "Old SPI" set to the previous SPI, the "New SPI" set to the desired new SPI value for the incoming SA, and the Keymat Index desired. Optionally, the host may include a DIFFIEHELLMAN parameter for a new Diffie-Hellman key. The peer completes the request for rekey as is normally done for HIP rekeying, except that the new address is kept as UNVERIFIED until the UPDATE nonce challenge is received as described above.
• Mobility with single SA pair (peer-initiated rekey)
  A second variation of this basic mobility scenario covers the case in which the mobile host does not attempt to rekey the existing SAs, but the peer host decides to do so. This typically results in a four packet exchange, as shown in Figure 5. The initial UPDATE packet from the mobile host is the same as in the scenario for which there is no rekey (Figure 3). The peer may decide to rekey, however, in which case the subsequent three packets follow the normal rekeying procedure described in the ESP specification [14], with the addition of the ECHO_REQUEST and ECHO_RESPONSE nonce for verification of the new address.

Global Mobility

Global mobility, defined as pre-communication mobility, is implemented in HIP by using the RVS mechanism discussed in section A.4.3.

The mobile host must always keep his RVS up to date on his current location. This way a host wanting to communicate with the mobile host can always sent the H1 packet directly to the RVS, abstracting away from the mobile host's locator.

A.4.6 Integration with DNS

An initiator willing to establish a HIP association with a responder served by a RVS would typically initiate a HIP exchange by sending an H1 towards the RVS IP address rather than towards the responder IP address (see section A.4.3). Consequently, we need a means to translate a domain name into the rendezvous server's domain name.

HIP introduces a new HIP DNS Resource Record to store Rendezvous Server (RVS), Host Identity (HI) and Host Identity Tag (HIT) information.

Simple static singly homed end-host

A HIP node (R) with a single static network attachment, wishing to be reachable by reference to its Fully Qualified Domain Name (www.example.com), would store in the DNS, in addition to its IP address(es) (IP-R), its Host Identity (HI-R) and Host Identity Tag (HIT-R) in a HIP resource record.

An initiator willing to associate with a node would typically issue the following queries:

QNAME=www.example.com, QTYPE=HIP

(QCLASS=IN is assumed and omitted from the examples)

Which returns a DNS packet with RCODE=0 and one or more HIP RRs with the HIT and HI (e.g. HIT-R and HI-R) of the responder in the answer section, but no RVS.

QNAME=www.example.com, QTYPE=A

Which returns a DNS packet with RCODE=0 and one or more A or AAAA RRs containing IP address(es) of the responder (e.g. IP-R) in the answer section.

78
The initiator would then send an I1 to the responder's IP addresses (IP-R), starting the Base Exchange.

Integration with DNS using a RVS

A mobile HIP node (R) wishing to be reachable by reference to its FQDN (www.example.com) would store in the DNS, possibly in addition to its IP address(es) (IP-R), its HI (HI-R), HIT (HIT-R) and the domain name(s) of its rendezvous server(s) (rsv.example.com) in HIP resource record(s). The mobile HIP node also needs to notify its rendezvous servers of any change in its set of IP address(es).

An initiator willing to associate with such mobile node would typically issue the following queries:

QNAME=www.example.com, QTYPE=HIP

Which returns a DNS packet with RCODE=0 and one or more HIP RRs with the HIT, HI and RVS domain name(s) (e.g. HIT-R, HI-R, and rsv.example.com) of the responder in the answer section.

QNAME=rsv.example.com, QTYPE=A

Which returns a DNS packet with RCODE=0 and one or more A or AAAA RRs containing IP address(es) of the responder's RVS (e.g. IP-RVS) in the answer section.

The initiator would then send an I1 to the RVS IP address (IP-RVS.) Following, the RVS will relay the I1 up to the mobile node's IP address (IP-R), which will complete the HIP exchange.

HIP Resource Record (HIP RR) Storage Format

The RDATA for a HIP RR consists of a public key algorithm type, the HIT length, a HIT, a public key, and optionally one or more rendezvous server(s).
Figure A.4: State Diagram
Figure A.5: Base Exchange with RVS

Figure A.6: Registration with the RVS

Figure A.7: Relay of the II packet

Figure A.8: Mobility procedure - No rekeying

81
Figure A.9: Mobility procedure - Rekey initiated by Mobile Host

Figure A.10: Mobility procedure - Rekey initiated by Peer Host

Figure A.11: Example of DNS integration with HIP
Figure A.12: Example of DNS integration with HIP using RVS
Two-step HIP Resolution

The resolution mechanism explained above is the currently implemented and normalized in the related Drafts [21]. However, this mechanism arises some problems:

- Creates dependency between HIP and DNS
- Makes impossible to communicate with a peer using HITs only because there’s no resolution mechanism mapping Host Identities to IPs
- There’s no reverse lookup mechanism
- Makes impossible to communicate with DNS using HITs, what could be useful for fiability purposes.

In [7] it’s proposed a new resolution mechanism, using two steps to accomplish the resolution of FQDNs to HIs with some more advantages:

This mechanism on fig. A.13 presents the advantage of allowing the resolution of HIs to IP, thus permitting the use of HIs (more likely, HITs) in APIs. Nevertheless, this mechanism also has its disadvantages:

- Introduces a second global resolution service
* Because HITs have no structure (unlike FQDNs), it would be difficult to implement a scalable resolution structure (Resolution services using DHT servers are being considered).

A.4.7 Multihoming

In this section we define the concept of generalization of an address called locator. A locator specifies a point-of-attachment to the network.

Host Multihoming

A (mobile or stationary) host may sometimes have more than one interface or global address. The host may notify the peer host of the additional interface or address by using the LOCATOR parameter. To avoid problems with the ESP anti-replay window**, a host should use a different SA for each interface or address used to receive packets from the peer host.

When more than one locator is provided to the peer host, the host should indicate which locator is preferred. By default, the addresses used in the base exchange are preferred until indicated otherwise.

Although the protocol may allow for configurations in which there is an asymmetric number of SAs between the hosts (e.g., one host has two interfaces and two inbound SAs, while the peer has one interface and one inbound SA), it is recommended that inbound and outbound SAs be created pairwise between hosts. When an ESP_INFO arrives to rekey a particular outbound SA, the corresponding inbound SA should be also rekeyed at that time. Although asymmetric SA configurations might be experimented with, their usage may constrain interoperability at this time. However, it is recommended that implementations attempt to support peers that prefer to use non-paired SAs. It is expected that this behavior will be modified in future revisions of this
protocol, once the issue and its implications are better understood.

**Communicating a new local address**

Consider the case between two single-homed hosts, in which one of the host notifies the peer of an additional address. It is recommended[**I-D.ietf-hip-base**] that the host set up a new SA pair for use on this new address. To do this, the multihomed host sends a LOCATOR with an ESP_INFO, indicating the request for a new SA by setting the "Old SPI" value to zero, and the "New SPI" value to the newly created incoming SPI. A Locator Type of "1" is used to associate the new address with the new SPI. The LOCATOR parameter also contains a second Type 1 locator: that of the original address and SPI. To simplify parameter processing and avoid explicit protocol extensions to remove locators, each LOCATOR parameter must list all locators in use on a connection (a complete listing of inbound locators and SPIs for the host). The multihomed host transitions to state REKEYING, waiting for an ESP_INFO (new outbound SA) from the peer and an ACK of its own UPDATE. As in the mobility case, the peer host must perform an address verification before putting the new address into active use. Figure 6 illustrates the basic packet exchange.

![Diagram](image)

Figure A.14: Basic Multihoming scenario

When processing inbound LOCATORs that establish new security associations on an interface with multiple addresses, a host uses the destination address of the UPDATE containing LOCATOR as the local address to which the LOCATOR plus ESP_INFO is targeted. Hosts may send UPDATEs with the same IP address in the LOCATOR to different peer addresses. This has the effect of creating multiple inbound SAs implicitly affiliated with different peer source addresses.

**Site Multihoming**

A host may have an interface that has multiple globally reachable IP addresses. Such a situation may be a result of the site having multiple upper Internet Service Providers, or just because the site provides all hosts with both IPv4 and IPv6 addresses. It is desirable that the host can stay reachable with all or any subset of the currently available globally routable addresses, independent on how they are provided. Note that a single interface may experience site multihoming while the host itself may have multiple interfaces. Note also that a host may be multi-homed and mobile simultaneously, and that a multi-homed host may want to protect the location of some of its interfaces while revealing the real IP address of some others.
Currently there aren’t any specify additional site multihoming extensions to HIP. Further alignment with the IETF shim6 working group may be considered in the future.

**Combined mobility and multihoming**

It looks likely that in the future many mobile hosts will be simultaneously mobile and multi-homed, *i.e.*, have multiple mobile interfaces. Furthermore, if the interfaces use different access technologies, it is fairly likely that one of the interfaces may appear stable (retain its current IP address) while some other(s) may experience mobility (undergo IP address change).

The use of LOCATOR plus ESP INFO should be flexible enough to handle most such scenarios, although more complicated scenarios have not been studied so far.

**A.5 Conclusions**

HIP presents itself as a promising technology. It is based on the paradigm of indirection architecture which has been a common approach for the current efforts to separate the identifiers from the locator.

HIP decouples the internetworking layer from the transport layer, allowing each to evolve separately. The decoupling makes end-host mobility and multi-homing easier, also across IPv4 and IPv6 networks. HIs make network renumbering easier, and they also make process migration and clustered servers easier to implement. Furthermore, being cryptographic in nature, they provide the basis for solving the security problems related to end-host mobility and multi-homing. As disadvantages, HIP has many functional requirements, particularly alterations to existing infrastructures and the introduction of new ones.

HIP is still not a complete solution as MIP, but this could be simply because of MIP’s head start. HIP stands on good premises and seems to have a lot of potential to evolve.

This manual is valid at the time of its elaboration. HIP and its implementations are in a development fase, and thus, may present alterations in the near future. Namely, HIP’s approach to large scale deployment infrastructure and scalability issues.
Appendix B

Log files

B.1 HIP Basic Base Exchange

B.1.1 II- HIP Initiator Packet

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>54</td>
<td>2006-03-01 10:51:02.629053</td>
<td>172.16.0.147</td>
<td>172.16.0.102</td>
<td>HIP I (HIP Initiator Packet)</td>
</tr>
</tbody>
</table>

Frame 54 (74 bytes on wire, 74 bytes captured)
Arrival Time: Mar 1, 2006 10:51:02.629053000
Time delta from previous packet: 10.098325000 seconds
Time since reference or first frame: 66.943953000 seconds
Frame Number: 54
Packet Length: 74 bytes
Capture Length: 74 bytes
Protocols in frame: eth:hip

Destination: EdimaxTe_55:67:ff (00:0e:2e:55:67:ff)
Source: UnivillC_29:4e:32 (00:03:0d:29:4e:32)
Type: IP (0x0800)

Internet Protocol, Src: 172.16.0.147 (172.16.0.147), Dst: 172.16.0.102 (172.16.0.102)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCH 0x00: Default; ECH: 0x00)
0000 00... = Differentiated Services Codepoint: Default (0x00)
..... ...0 = ECH-Capable Transport (ECT): 0
..... ...0 = ECH-CE: 0
Total Length: 60
Identification: 0xb31 (48433)
Flags: 0x04 (Don't Fragment)
0... = Reserved bit: Not set
.1... = Don't fragment: Set
..0... = More fragments: Not set
Fragment offset: 0
Time to live: 64
Protocol: Unknown (0x63)
Header checksum: 0x2414 [correct]
Source: 172.16.0.147 (172.16.0.147)
Destination: 172.16.0.102 (172.16.0.102)

Host Identity Protocol
Payload Protocol: 59
Header Length: 4
Packet Type: 1
Version: 1, Reserved: 0
HIP Controls: 0x2400
  001. .... .... .... = Sender's HIT Type: Type 1 HIT (0x0001)
  ...0 01. .... .... = Destination's HIT Type: Type 1 HIT (0x0001)
  .... .... ....0. = Certificate (One or more CER packets follows): False
  .... .... .......0 = Anonymous (Sender's HI is anonymous): False
Checksum: 0x90b3 (correct)
Sender's HIT: 4054e3568fbad8d39808ec6e64f2f1415
Receiver's HIT: 4084d242b086f81bd0352d36a74af5d0
B.1.2 R1- HIP Responder Packet

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>55</td>
<td>2006-03-01 10:51:02.629340</td>
<td>172.16.0.102</td>
<td>172.16.0.147</td>
<td>HIP HIP R1 (HIP Responder Packet)</td>
</tr>
</tbody>
</table>

Frame 55 (634 bytes on wire, 634 bytes captured)

Arrival Time: Mar 1, 2006 10:51:02.629340000
Time delta from previous packet: 0.000287000 seconds
Time since reference or first frame: 66.944240000 seconds
Frame Number: 55
Packet Length: 634 bytes
Capture Length: 634 bytes

Protocols in frame: eth:ip:

Destination: UnwivllC_29:4e:32 (00:03:0d:29:4e:32)
Source: EdimaxTe_55:57:ff (00:0e:2e:55:57:ff)
Type: IP (0x0800)

Internet Protocol, Src: 172.16.0.102 (172.16.0.102), Dst: 172.16.0.147 (172.16.0.147)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DS: Default; ECN: 0x00)
0000 00... = Differentiated Services Codepoint: Default (0x00)
.... 0... = ECN-Capable Transport (ECT): 0
.... 0... = ECN-CE: 0
Total Length: 620
Identification: 0xc304 (49924)
Flags: 0x04 (Don't Fragment)
0... = Reserved bit: Not set
.1... = Don't fragment: Set
...0... = More fragments: Not set
Fragment offset: 0
Time to live: 64
Protocol: Unknown (0x63)
Header checksum: 0xc11 (correct)
Source: 172.16.0.102 (172.16.0.102)
Destination: 172.16.0.147 (172.16.0.147)

Host Identity Protocol
Payload Protocol: 59
Header Length: 74
Packet Type: 2
Version: 1, Reserved: 0

HIP Controls: 0x2400
001... ....... ....... = Sender's HIT Type: Type 1 HIT (0x0001)
.... 01... ....... = Destination's HIT Type: Type 1 HIT (0x0001)
.... ....... 0... = Certificate (One or more CER packets follows): False
.... ....... 0... = Anonymous (Sender's HI is anonymous): False

Checksum: 0x0f04b (correct)
Sender's HIT: 4064d242b086f81b00352d3567474f5dd
Receiver's HIT: 4054b366f8ad633980e8ce664fe24115

HIP Parameters
R1 COUNTER (type=128, length=12)
Reserved: 0x00000000
R1 Counter: 0x0000000000000072
PUZZLE (type=257, length=12)
Puzzle Difficulty: 39
Opaque Data: 0x0000
Puzzle Random I: 5b08db006b299b543
DIFFIE_HILLMAN (type=513, length=193)
3 (1936-bit MD5 group)
Public Value: 65188faa8c861c44211268d94c9b8a42488dc610d430a98bd...
HIP_TRANSFORM (type=577, length=12)
1 (AES-CBC with HMAC-SHA1)
2 (3DES-CBC with HMAC-SHA1)
3 (3DES-CBC with HMAC-MD5)
4 (BLOWFISH-CBC with HMAC-SHA1)
5 (NULL with HMAC-SHA1)
6 (NULL with HMAC-MD5)

ESP_TRANSFORM (type=4096, length=14)
Reserved: 0x0000
1 (AES-CBC with HMAC-SHA1)
2 (3DES-CBC with HMAC-SHA1)
3 (3DES-CBC with HMAC-MD5)
4 (BLOWFISH-CBC with HMAC-SHA1)
5 (NULL with HMAC-SHA1)
6 (NULL with HMAC-MD5)

HOST_ID (type=705, length=147)
Host Identity Length: 136
Domain Identifier Type: 1
Domain Identifier Length: 7
Host Identity flags: 0x02Dff05
0000 0010 0000 0010 ....... .... .... = Flags: key is associated with non-zone entity (0x00000002)
....... .... .... 1111 1111 .... = Protocol: key is valid for any protocol (0x000000ff)
....... .... .... 0000 0101 = Algorithm: RSA (0x00000005)

RSA Host Identity e_len (exponent length): 3
RSA Host Identity e (exponent): 010001
RSA Host Identity n (public modulus): E40F9A6688333DD276D66A0B8A7C8F2F844FC7FA065FED...

FQDN: ml-1024

HIP_SIGNATURE_2 (type=61639, length=129)
6 (RSA)
Signature: 46EE50653D280D6579F9ED2768C6F16BF9B0B97AC6723E062...
B.1.3 12- Second HIP Initiator Packet

Frame 56 (690 bytes on wire, 690 bytes captured)
Arrival Time: Mar 1, 2006 10:51:2.729900000
Time delta from previous packet: 0.1005660000 seconds
Time since reference or first frame: 67.0448000000 seconds
Frame Number: 56
Packet Length: 690 bytes
Capture Length: 690 bytes
Protocols in frame: eth:hip
Source: Univ11C_29:4e:32 (00:03:0d:29:4e:32)
Type: IP (0x0800)
Internet Protocol, Src: 172.16.0.147 (172.16.0.147), Dst: 172.16.0.102 (172.16.0.102)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
0000 00.. = Differentiated Services Codepoint: Default (0x00)
..... .0. = ECN-Capable Transport (ECT): 0
..... .0. = ECN-CE: 0
Total Length: 676
Identification: 0xb695 (48533)
Flags: 0x04 (Don't Fragment)
0.. = Reserved bit: Not set
1.. = Don't fragment: Set
...0 = More fragments: Not set
Fragment offset: 0
Time to live: 64
Protocol: Unknown (0x63)
Header checksum: 0x2148 [correct]
Source: 172.16.0.147 (172.16.0.147)
Destination: 172.16.0.102 (172.16.0.102)
Host Identity Protocol
Payload Protocol: 59
Header Length: 81
Packet Type: 3
Version: 1, Reserved: 0
HIP Controls: 0x2400
001. .... ...... = Sender's HIT Type: Type 1 HIT (0x0001)
....0 01. ...... = Destination's HIT Type: Type 1 HIT (0x0001)
..... ...... .0. = Certificate (One or more CER packets follows): False
..... ...... ....0 = Anonymous (Sender's HI is anonymous): False
Checksum: 0xb947 [correct]
Sender's HIT: 4064B356FBD9808BC56645EFE1418
Receiver's HIT: 40846242B086F81BD0352D36A74AF5ED
HIP Parameters
ESP INFO (type=65, length=12)
Reserved: 0x0000
Keymat Index: 0x0048
Old SPI: 0x00000000
New SPI: 0x175b3481
RI COUNTER (type=128, length=12)
Reserved: 0x00000000
RI Counter: 0x0000000000000072
SOLUTION (type=321, length=20)
Opaque Data: 0x0000
Puzzle Difficulty K: 10
Puzzle Lifetime: 39
Opaque Data: 0x0000
Puzzle Random 1: 5508DAE06928A543

92
Puzzle Solution J: 6EF70B12AFA25977
DIFFIE_HELLMAN (type=513, length=193)
  3 (1536-bit MODP group)
  Public Value: 912492335688E881587F40EAF33FA9440D664608A5D1A9...
HIP_TRANSFORM (type=577, length=2)
  1 (AES-CBC with HMAC-SHA1)
ESP_TRANSFORM (type=4095, length=4)
  Reserved: 0x0000
  1 (AES-CBC with HMAC-SHA1)
ENCRYPTED (type=641, length=180)
  Reserved: 0x00000000
  IV: DABF7C3E34D36043
  Encrypted Data (168 bytes)
HMAC (type=61505, length=20)
  HMAC: 6DA90552B9F0F401282CF878C80F489F96460F5
HIP_SIGNATURE (type=61697, length=129)
  5 (RSA)
  Signature: 24243E0B9D7494B72B5287D9F4996E4448B7812C58830D1...
B.1.4  R2- Second HIP Responder Packet

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>87</td>
<td>2006-03-01 10:51:02.755112 172.16.0.102 172.16.0.147 HIP</td>
<td>HIP R2 (Second HIP Responder Packet)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Frame 57 (250 bytes on wire, 250 bytes captured)
Arrival Time: Mar 1, 2006 10:51:02.755112000
Time delta from previous packet: 0.0252120000 seconds
Time since reference or first frame: 67.0700120000 seconds
Frame Number: 57
Packet Length: 250 bytes
Capture Length: 250 bytes
Protocols in frame: eth; hip

Destination: UnivialC_29:4e:32 (00:03:0d:29:4e:32)
Source: EdimaxTe_55:67:ff (00:0e:2e:55:67:ff)
Type: IP (0x0800)

Internet Protocol, Src: 172.16.0.102 (172.16.0.102), Dst: 172.16.0.147 (172.16.0.147)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN 0x00)
0000 00.. = Differentiated Services Codepoint: Default (0x00)
.... .0. = ECN-Capable Transport (ECT): 0
..... 0. = ECN-CE: 0
Total Length: 256
Identification: 0xc381 (50049)
Flags: 0x04 (Don't Fragment)
 0... = Reserved bit: Not set
 1.. = Don't fragment: Set
 0..0 = More fragments: Not set
Fragment offset: 0
Time to live: 64
Protocol: Unknown (0x63)
Header checksum: 0x101e [correct]
Source: 172.16.0.102 (172.16.0.102)
Destination: 172.16.0.147 (172.16.0.147)

Host Identity Protocol
Payload Protocol: 59
Header Length: 26
Packet Type: 4
Version: 1, Reserved: 0
HIP Controls: 0x2400
001. 001. 001. = Sender's HIT Type: Type 1 HIT (0x0001)
 001. 001. 001. = Destination's HIT Type: Type 1 HIT (0x0001)
............ 0. = Certificate (one or more CER packets follows): False
............ 0. = Anonymous (Sender's HI is anonymous): False
Checksum: 0x55de (correct)
Sender's HIT: 4084D242B086FB61BD0352D3847444F5BD
Receiver's HIT: 4054B356F84D29398062E644FEF1415

HIP Parameters
ESP INFO (type=65, length=12)
Reserved: 0x0000
Key Index: 0x0048
Old SPI: 0x00000000
New SPI: 0x21f3c602
HMAC_2 (type=61569, length=20)
HMAC: A57D4684874B3069608D1E882D6DF4304469930F
HIP_SIGATURE (type=61697, length=129)
S (554)
Signature: 4AC139681A1191EB3F2934A45F67D7C020E37FC490264B16...
B.2 Mobility
### B.2.1 First HIP Update Packet

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>371</td>
<td>2006-03-09 12:10:38:320567</td>
<td>172.16.100.1</td>
<td>172.16.0.102</td>
<td>HIP UPDATE (HIP Update Packet)</td>
</tr>
</tbody>
</table>

**Frame 371** (290 bytes on wire, 290 bytes captured)
- Arrival Time: Mar 9, 2006 12:10:38.320567000
- Time delta from previous packet: 33.532780000 seconds
- Time since reference or first frame: 80.476755000 seconds
- Frame Number: 371
- Frame Length: 290 bytes
- Capture Length: 290 bytes
- Protocols in frame: ethip:hip

- Destination: EdimaxTE_55:67:76f5 (00:0e:2e:55:67:76f5)
- Source: UniWILL-29:4e:32 (00:03:04:29:4e:32)
- Type: IP (0x0800)

**Internet Protocol**, Src: 172.16.100.1 (172.16.100.1), Dst: 172.16.0.102 (172.16.0.102)
- Version: 4
- Header Length: 20 bytes
- Differentiated Services Field: 0x00 (DS00: Default; ECN: 0x00)
  - 0000 00.. = Differentiated Services Codepoint: Default (0x00)
  - .... 0. = ECN-Capable Transport (ECT): 0
  - .... 0. = ECN-CE: 0
- Total Length: 276
- Identification: 0xf519 (62745)
- Flags: 0x04 (Don’t Fragment)
  - 0... = Reserved bit: Not set
  - ..1. = Don’t fragment: Set
  - ...0. = More fragments: Not set
- Fragment offset: 0
- Time to live: 64
- Protocol: Unknown (0x63)
- Header checksum: 0xB765 [correct]
- Source: 172.16.100.1 (172.16.100.1)
- Destination: 172.16.0.102 (172.16.0.102)

**Host Identity Protocol**
- Payload Protocol: 59
- Header Length: 31
- Packet Type: 6
- Version: 1, Reserved: 0

- HIP Controls: 0x2400
  - 001. .... .... = Sender’s HIT Type: Type 1 HIT (0x0001)
  - .... 01. .... = Destination’s HIT Type: Type 1 HIT (0x0001)
  - .... .... 0. = Certificate (One or more CER packets follows): False
  - .... .... .... 0 = Anonymous (Sender’s HI is anonymous): False

- Checksum: 0x6a93 [correct]

- HIP Parameters
  - LOCATOR (type=193, length=28)
    - Traffic Type: 0
    - Locator Type: 1
    - Locator Length: 5
    - Reserved: 0x1
    - Locator Lifetime: 0x00000000
    - Address: 34:3e:66:8::ff
  - ESP (type=85, length=12)
    - Reserved: 0x0000
    - Keymat Index: 0x0090
    - Old SPI: 0x2fa3d90e
    - New SPI: 0x24a3b68e
    - SEQ (type=385, length=4)

96
Update ID: 0x00000001
HMAC (type=61505, length=20)
HMAC: E415E5453F5C4DF8A488A54763A9A9F17D1B6E1
HIP_SIGNATURE (type=61697, length=129)
S (RSA)
Signature: 3AE1E113CE26E32D2F52CAF0AE00F56747CC40FD662BDE...
B.2.2 Second HIP Update Packet

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
<th>HIP UPDATE (HIP Update Packet)</th>
</tr>
</thead>
<tbody>
<tr>
<td>372</td>
<td>2006-03-09 12:10:38.323721</td>
<td>172.16.0.102</td>
<td>172.16.100.1</td>
<td>HIP</td>
<td></td>
</tr>
</tbody>
</table>

Frame 372 (274 bytes on wire, 274 bytes captured)
Arrival Time: Mar 9, 2006 12:10:38.323721000
Time delta from previous packet: 0.0031644000 seconds
Time since reference or first frame: 80.4799590000 seconds
Frame Number: 372
Packet Length: 274 bytes
Capture Length: 274 bytes
Protocols in frame: ether:hip

Destination: UniwillIC_29:4e:32 (00:03:0d:29:4e:32)
Type: IP (0x0800)

Internet Protocol, Src: 172.16.0.102 (172.16.0.102), Dst: 172.16.100.1 (172.16.100.1)
Version: 4
Header Length: 20 bytes
Differentiated Services Field: 0x00 (DSCH 0x00: Default; ECN: 0x00)
    0000 00.. = Differentiated Services Codepoint: Default (0x00)
    ..... ... = ECN-Category (학적): 0
    ..... ..... = ECN-CE: 0
Total Length: 260
Identification: 0x1f8a (6314)
Flags: 0x04 (Don't Fragment)
    0... = Reserved bit: Not set
    .1.. = Don't fragment: Set
    ...0 = More fragments: Not set
Fragment offset: 0
Time to live: 64
Protocol: Unknown (0x63)
Header checksum: 0x8465 [correct]
Source: 172.16.0.102 (172.16.0.102)
Destination: 172.16.100.1 (172.16.100.1)

Host Identity Protocol
Payload Protocol: 59
Header Length: 29
Packet Type: 6
Version: 1, Reserved: 0
HIP Controls: 0x2400
    001... = Sender's HIT Type: Type 1 HIT (0x0001)
    ..0 01... = Destination's HIT Type: Type 1 HIT (0x0001)
    ..... ...0 = Certificate (One or more CES packets follow): False
    ..... .....0 = Anonymous (Sender's HI is anonymous): False
Checks: 0x81f3d (correct)
Sender's HIT: 4084d242b088f81b937932d36474abf5dd
Receiver's HIT: 4064b3b6f8bab39808ece364bef1445

HIP Parameters
ESP INFO (type=65, length=12)
    Reserved: 0x0000
    Keymat Index: 0x0000
    Old SPI: 0x204222f3
    New SPI: 0x2d112ee
SEQ (type=385, length=4)
    Update ID: 0x00000001
ACK (type=449, length=4)
    ACKed Peer Update ID: 0x00000001
HMAC (type=61505, length=20)
    HMAC: 657a9278c9740b41e8632997b5d1223e92b5f54
HIP_SIGNATURE (type=61697, length=129)
    E (RSK)
Signature: 9B79081EBE3E4FCE71EB0E991A25A05539EF03D8EBC21302B...
ECHO_REQUEST (No sig.) (type=63061, length=4)
Opaque Data: 7F98BC5B
B.2.3 Third HIP Update Packet

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>373</td>
<td>2006-03-09 12:10:38.335363</td>
<td>172.16.100.1</td>
<td>172.16.0.102</td>
<td>HIP</td>
</tr>
<tr>
<td></td>
<td>Frame 373 (250 bytes on wire, 250 bytes captured)</td>
<td></td>
<td></td>
<td>HIP UPDATE (HIP Update Packet)</td>
</tr>
<tr>
<td></td>
<td>Arrival Time: Mar 9, 2006 12:10:38.335363000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Time delta from previous packet: 0.0126420000 seconds</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Time since reference or first frame: 80.4926010000 seconds</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Frame Number: 373</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Packet Length: 250 bytes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Capture Length: 250 bytes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Protocols in frame: eth:hip</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Source: UnviLlC_29:4e:32 (00:03:0d:29:4e:32)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Type: IP (0x0800)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Internet Protocol, Src: 172.16.100.1 (172.16.100.1), Dst: 172.16.0.102 (172.16.0.102)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Version: 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Header length: 20 bytes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Differentiated Services Field: 0x00 (DSCH 0x00: Default; ECN: 0x00)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0000 00.. = Differentiated Services Codepoint: Default (0x00)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>= ECN-Capable Transport (ECT): 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>= ECN-CE: 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total Length: 236</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Identification: 0xf529 (62761)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Flags: 0x04 (Don't Fragment)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>= Reserved bit: Not set</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>= Don't Fragment: Set</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>= More fragments: Not set</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fragment offset: 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Time to live: 64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Protocol: Unknown (0x63)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Header checksum: 0x87fd [correct]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Source: 172.16.100.1 (172.16.100.1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Destination: 172.16.0.102 (172.16.0.102)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Host Identity Protocol</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Payload Protocol: 69</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Header Length: 26</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Packet Type: 6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Version: 1, Reserved: 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HIP Controls: 0x2400</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>001. ..... . = Sender's HIT Type: Type 1 HIT (0x0001)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>= Destination HIT Type: Type 1 HIT (0x0001)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>= Certificate (One or more CER packets follows): False</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>= Anonymous (Sender's HI is anonymous): False</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Checksum: 0x9b71 [correct]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sender's HIT: 40543558F8B4ADB398603EEED64EFFF1415</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Receiver's HIT: 4084D242B086FB1ED3525236A74A655DD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HIP Parameters</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ACK (type=448, length=4)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ACKed Peer Update ID: 0x00000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HMAC (type=61805, length=20)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HMAC: 2DE7DCF083CDD116B9064799C617BF72D764F5540</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HIP_SIGNATURE (type=61697, length=129)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>S (RSA)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Signature: 71D4CD3AD8FF663B81EA294E5D1FED1CA90A049B8E7081E...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ECHO_RESPONSE (No sig.) (type=63425, length=4)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Opaque Data: 7F98BC58</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
B.3 Base Exchange with RVS

B.3.1 HIP Initiator Packet

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>331</td>
<td>2006-03-08</td>
<td>17:51:14.978694</td>
<td>172.16.0.147</td>
<td>HIP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>172.16.0.102</td>
<td>HIP II (HIP Initiator Packet)</td>
</tr>
</tbody>
</table>

Frame 331 (74 bytes on wire, 74 bytes captured)
Arrival Time: Mar 8, 2006 17:51:14.978694000
Time delta from previous packet: 92.790064000 seconds
Time since reference or first frame: 92.790064000 seconds
Frame Number: 331
Packet Length: 74 bytes
Capture Length: 74 bytes
Protocols in frame: e:ethr:hip

Source: UnivillC-29:4e:32 (00:03:0d:29:4e:32)
Type: IP (0x0800)

Internet Protocol, Src: 172.16.0.147 (172.16.0.147), Dst: 172.16.0.102 (172.16.0.102)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
0000 00... = Differentiated Services Codepoint: Default (0x00)
.... 0... = ECN-Capable Transport (ECT): 0
.... 0... = ECN-CE: 0
Total Length: 60
Identification: 0x8f20 (28448)
Flags: 0x04 (Don't Fragment)
0... = Reserved bit: Not set
1... = Don't fragment: Set
0... = More fragments: Not set
Fragment offset: 0
Time to live: 64
Protocol: Unknown (0x53)
Header checksum: 0x7225 [correct]
Source: 172.16.0.147 (172.16.0.147)
Destination: 172.16.0.102 (172.16.0.102)

Host Identity Protocol
Payload Protocol: 59
Header Length: 4
Packet Type: 1
Version: 1, Reserved: 0
HIP Controls: 0x2400
001... 001... 000... = HIP's HIT Type: Type 1 HIT (0x0001)
.... 01... 01... = Destination's HIT Type: Type 1 HIT (0x0001)
.... 00... 00... = Certificate (One or more CER packets follows): False
.... 00... 00... = Anonymous (Sender's HI is anonymous): False
Checksum: 0x0000 (correct)
Sender's HIT: 4054b355bf4add3980b0ec664f8f1415
Receiver's HIT: 4055e7ed1a941d9242443c3f93935e
B.3.2 RI- HIP Responder Packet

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>332</td>
<td>2006-03-08</td>
<td>17:51:14.977939</td>
<td>172.16.10.10</td>
<td>172.16.0.147</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HIP</td>
<td>HIP</td>
<td>HIP R1 (HIP Responder Packet)</td>
</tr>
</tbody>
</table>

Frame 322 (662 bytes on wire, 662 bytes captured)
Arrival Time: Mar 8, 2006 17:51:14.977939000
Time delta from previous packet: 0.00126450000 seconds
Time since reference or first frame: 92.791309000 seconds
Frame Number: 332
Packet Length: 662 bytes
Capture Length: 662 bytes
Protocols in frame: eth:ip:hip

Destination: UnivillC_29:4e:32 (00:03:0d:29:4e:32)
Source: AusrtekC_03:23:46 (00:13:d4:03:23:46)
Type: IP (0x0800)

Internet Protocol, Src: 172.16.10.10 (172.16.10.10), Dst: 172.16.0.147 (172.16.0.147)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DS0 0x00: Default; ECN: 0x00)
0x0000 = Differentiated Services Codepoint: Default (0x00)
... .0. = ECN-Capable Transport (ECT): 0
... .0. = ECN-CE: 0
Total Length: 64B
Identification: 0x2606 (9734)
Flags: 0x04 (Don't Fragment)
0... = Reserved bit: Not set
1... = Don't fragment: Set
..0. = More fragments: Not set
Fragment offset: 0
Time to live: 64
Protocol: Unknown (0x63)
Header checksum: 0x244f [correct]
Source: 172.16.10.10 (172.16.10.10)
Destination: 172.16.0.147 (172.16.0.147)
Host Identity Protocol
Payload Protocol: 59
Header Length: 77
Packet Type: 2
Version: 1, Reserved: 0
HIP Controls: 0x2400
001. ...... .... = Sender's HIT Type: Type 1 HIT (0x0001)
...0 01. ...... .... = Destination's HIT Type: Type 1 HIT (0x0001)
...... ...... ..0. = Certificate (One or more CER packets follows): False
...... ...... ....0. = Anonymous (Sender's HI is anonymous): False
Checksum: 0x6448 (Incorrect, should be 0x64c8)
Sender's HIT: 4055E7E0D1A941D56242443C5FC95338E
Receiver's HIT: 4054B358FBAD339808ECE664F6F1415
HIP Parameters
RI COUNTER (type=128, length=12)
Reserved: 0x0000000000
RI Counter: 0x0000000000000000
PUZZLE (type=257, length=12)
 Puzzle Difficulty K: 10
 Puzzle Lifetime: 39
 Opaque Data: 0x0000
 Puzzle Random I: 313E4287F964E48A
DIFFIE_HELLMAN (type=813, length=193)
3 (1536-bit MDP group)
 Public Value: 2581A4460F5452C4CCB26769D32290076FC708184FF2D22A...
HIP_TRANSFORM (type=577, length=12)
1 (AES-CBC with HMAC-SHA1)
2 (DES-CBC with HMAC-SHA1)
3 (3DES-CBC with HMAC-MD5)
4 (Blowfish-CBC with HMAC-SHA1)
5 (NULL with HMAC-SHA1)
6 (NULL with HMAC-MD5)

ESP_TRANSFORM (type=4095, length=14)
Reserved: 0x0000
1 (AES-CBC with HMAC-SHA1)
2 (DES-CBC with HMAC-SHA1)
3 (3DES-CBC with HMAC-MD5)
4 (Blowfish-CBC with HMAC-SHA1)
5 (NULL with HMAC-SHA1)
6 (NULL with HMAC-MD5)

HUST_ID (type=705, length=153)
Host Identity Length: 136
Domain Identifier Type: 1
Domain Identifier Length: 13
Host Identity flags: 0x0202f005
0000 0010 0000 0010 ..... ..... ..... = Flags: key is associated with non-zone entity (0x00002002)
..... ..... ..... 1111 1111 ..... = Protocol: key is valid for any protocol (0x000000ff)
..... ..... ..... 0000 0101 = Algorithm: RSA (0x00000005)
RSA Host Identity e_len (exponent length): 3
RSA Host Identity e (exponent): 010001
RSA Host Identity n (public modulus): E4AEEAA2D3125F412E54CF636D8161C76C49E7B7BE18719...
FUDR: terrance-1024

HIP_SIGNATURE_2 (type=61633, length=129)
5 (RSA)
Signature: 11D20176269845834034A4F2D2F9E86648D103EA2EC3C1193...
B.3.3 12- Second HIP Initiator Packet

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>333</td>
<td>2006-03-08 17:51:15.133924</td>
<td>172.16.0.147</td>
<td>172.16.10.10</td>
<td>HIP HIP 12 (Second HIP Initiator Packet)</td>
</tr>
</tbody>
</table>

Frame 333 (690 bytes on wire, 690 bytes captured)
Arrival Time: Mar 8, 2006 17:51:15.1339240000
Time delta from previous packet: 0.1559800000 seconds
Time since reference or first frame: 92.9472940000 seconds
Frame Number: 333
Packet Length: 690 bytes
Capture Length: 690 bytes
Protocols in frame: eth:ip:hip

Source: UniwilLC_29:4e:32 (00:03:04:29:4e:32)
Type: IP (0x0800)

Internet Protocol, Src: 172.16.0.147 (172.16.0.147), Dst: 172.16.10.10 (172.16.10.10)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00; Default; ECN: 0x00)
0000 00.. = Differentiated Services Codepoint: Default (0x00)
..... 0.. = ECN-Capable Transport (ECT): 0
...... 0.. = ECN-CE: 0
Total Length: 676
Identification: 0x66fbd (28605)
Flags: 0x04 (Don't Fragment)
0.. = Reserved bit: Not set
1.. = Don't fragment: Set
..... = More fragments: Not set
Fragment offset: 0
Time to live: 64
Protocol: Unknown (0x63)
Header checksum: 0x857c (correct)
Source: 172.16.0.147 (172.16.0.147)
Destination: 172.16.10.10 (172.16.10.10)

Host Identity Protocol
Payload Protocol: 59
Header Length: 81
Packet Type: 3
Version: 1, Reserved: 0
HIP Controls: 0x2400
001. ....... 0001.. = Sender's HIT Type: Type 1 HIT (0x0001)
001. ....... 0001.. = Destination's HIT Type: Type 1 HIT (0x0001)
0000......0 = Certificate (One or more CER packets follows): False
0000......0 = Anonymous (Sender's HI is anonymous): False
Checksum: 0xada1 (correct)
Sender's HIT: 4054B958FBAD8D3980E564FEF1415
Receiver's HIT: 405SEC7E6D1A941D59242445C3FC9935E

HIP Parameters
ESP INFO (type=65, length=12)
Reserved: 0x0000
Repeat Index: 0x0048
Old SPI: 0x00000000
New SPI: 0x7f0f017e7
R1 COUNTER (type=126, length=12)
Reserved: 0x00000000
R1 Counter: 0000000000000000C
SOLUTION (type=321, length=20)
Puzzle Difficulty K: 10
Puzzle Lifetime: 39
Opaque Data: 0x0000
Puzzle Random I: 313E2487F9E64F8A

104
Puzzle Solution J: B0EC0677A345CCA
DIFFIE_HELLMAN (type=515, length=193)
3 (1536-bit MODP group)
Public Value: 86503A4613BA2099C010BEEA3692874C1401BA3CED2D348EFC...
HIP_TRANSFORM (type=577, length=2)
1 (AES-CBC with HMAC-SHA1)
ESP_TRANSFORM (type=4095, length=4)
Reserved: 0x0000
1 (AES-CBC with HMAC-SHA1)
ENCRYPTED (type=641, length=180)
Reserved: 0x00000000
IV: B795C308C342B86
Encrypted Data (168 bytes)
HMAC (type=61505, length=20)
HMAC: 4D7E42B84ED4EEB1E40889C27204486CF9440333
HIP_SIGNATURE (type=61697, length=129)
5 (RSA)
Signature: 8BEAD6E98A39D8FF0181AD8184567A1F7C5456291DCC8A8...
B.3.4 R2-Second HIP Responder Packet

No. Time Source Destination Protocol Info
334 2006-03-08 17:51:15.179413 172.16.10.10 172.16.0.147 HIP R2 (Second HIP Responder Packet)

Frame 334 (250 bytes on wire, 250 bytes captured)
Arrival Time: Mar 8, 2006 17:51:15.179413000
Time delta from previous packet: 0.045489000 seconds
Time since reference or first frame: 92.992785000 seconds
Frame Number: 334
Packet Length: 250 bytes
Capture Length: 250 bytes
Protocols in frame: eth:ip:hip

Destination: Univill1C_29:4e:32 (00:03:0d:29:4e:32)
Source: AsustekC_03:23:46 (00:13:4d:03:23:46)
Type: IP (0x0800)

Internet Protocol, Src: 172.16.10.10 (172.16.10.10), Dest: 172.16.0.147 (172.16.0.147)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DS00 DS00: Default; ECN: 0x00)
0000 00.. = Differentiated Services Codepoint: Default (0x00)
...... 0.. = ECN-Capable Transport (ECT): 0
...... 0.. = ECN-CE: 0
Total Length: 236
Identification: 0x249d (9936)
Flags: 0x04 (Don't Fragment)
0... = Reserved bit: Not set
1.. = Don't fragment: Set
.0. = More fragments: Not set
Fragment offset: 0
Time to live: 64
Protocol: Unknown (0x63)
Header checksum: 0xb021 [correct]
Source: 172.16.10.10 (172.16.10.10)
Destination: 172.16.0.147 (172.16.0.147)

Host Identity Protocol
Payload Protocol: 59
Header Length: 26
Packet Type: 4
Version: 1, Reserved: 0
HIP Controls: 0x2400
001. ........ .... = Sender's HIT Type: Type 1 HIT (0x0001)
...... 01... ...... = Destination's HIT Type: Type 1 HIT (0x0001)
...... ...... ...... = Certificate (One or more CER packets follows): False
...... ...... ...... = Anonymous (Sender's HI is anonymous): False
Checksum: 0x5b68 (correct)
Sender's HIT: 4055F7ED1A941D582424433CF9933E
Receiver's HIT: 4056B368FB4BD9330ECE26647EF1415

HIP Parameters
ESP INFO (type=66, length=12)
Reserved: 0x0000
Keymap Index: 0x0048
Old SPI: 0x00000000
New SPI: 0x9c9b5537
HMAC_2 (type=6169, length=20)
HMAC: BAEB44D0E9E9299DD9294C19DCC03A171D597A
HIP_SIGNATURE (type=6169, length=129)
S (RSA)
Signature: 4D791ABCA377C08FF6A3D478394763307C0A1D7EC9212EC
B.3.5 SIP - INVITE Request

<table>
<thead>
<tr>
<th>No.</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>14:36:28.506580</td>
<td>172.16.0.12</td>
<td>SIP/SDP Request: INVITE sip:Server, with session description</td>
</tr>
</tbody>
</table>

Frame 25 (771 bytes on wire, 771 bytes captured)
Arrival Time: Jul 1, 2006 14:36:28.506580000
Time delta from previous packet: 0.488560000 seconds
Time since reference or first frame: 28.309080000 seconds
Frame Number: 25
Packet Length: 771 bytes
Capture Length: 771 bytes

Ethernet II, Src: Uniwilc_29:4e:32 (00:03:0d:29:4e:32), Dst: EdimaxTe_54:54:7e (00:0e:2e:54:54:7e)
Destination: EdimaxTe_54:54:7e (00:0e:2e:54:54:7e)
Source: Uniwilc_29:4e:32 (00:03:0d:29:4e:32)
Type: IP (0x0800)

Internet Protocol, Src: 172.16.0.200 (172.16.0.200), Dst: 172.16.0.12 (172.16.0.12)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSAP 0x00: Default; ECN: 0x00)
0000 00.. = Differentiated Services Codepoint: Default (0x00)
.... ...0 = ECN-Capable Transport (ECT): 0
.... ....0 = ECN-CE: 0
Total Length: 787
Identification: 0x0001 (1)
Flags: 0x04 (Don't Fragment)
0... = Reserved bit: Not set
..1. = Don't fragment: Set
...0. = More fragments: Not set
Fragment offset: 0
Time to live: 64
Protocol: UDP (0x11)
Header checksum: 0xd0f2 [correct]
Source: 172.16.0.200 (172.16.0.200)
Destination: 172.16.0.12 (172.16.0.12)

User Datagram Protocol, Src Port: sip (5060), Dst Port: 5070 (5070)
Source port: sip (5060)
Destination port: 5070 (5070)
Length: 737
Checksum: 0x5be7 [incorrect, should be 0x9d59]

Session Initiation Protocol
Request-Line: INVITE sip:Server SIP/2.0
Method: INVITE
Request-URI: True
Suspected resend of frame: 23

Message Header
Call-ID: deb6cfc33c1ecb67c0a1b1945e580768f6172.16.0.200
CSeq: 1 INVITE
From: "Client" <sip:Client@tag=12345
SIP Display info: "Client"
SIP from address: sip:Client
SIP tag: 12345
To: "Server" <sip:Server>
SIP Display info: "Server"
SIP to address: sip:Server
Via: SIP/2.0/UDP 172.16.0.200:5060;branch=z9h4b5f293a5f543c03791401be8ab7c5153
Max-Forwards: 70
Contact: "Client Name" <sip:Client:5060>
Contact Binding: "Client Name" <sip:Client:5060>
URI: "Client Name" <sip:Client:5060>
SIP Display info: "Client Name"
SIP contact address: sip:Client:5060
My-Header: my header value
Content-Type: application/sdp
My-Other-Header: my new header value
Call-Info: <http://www.ntt.nist.gov>
Content-Length: 262

Message body

Session Description Protocol

Session Description Protocol Version (v): 0
Owner/Creator, Session Id (o): 4855 137607999569569568020 137607999569569568020 IN IP4 172.16.0.200
Owner Username: 4855
Session ID: 137607999569569568020
Session Version: 137607999569569568020
Owner Network Type: IN
Owner Address Type: IP4
Owner Address: 172.16.0.200
Session Name (a): mysession session
Session Information (i): Subarul.mov
Phone Number (p): +66 8 32018010
Connection Information (c): IN IP4 129.6.55.78
Connection Network Type: IN
Connection Address Type: IP4
Connection Address: 129.6.55.78
Time Description, active time (t): 0 0
Session Start Time: 0
Session Stop Time: 0
Media Description, name and address (m): video 12344 RTP/AVP 0 4 18
Media Type: video
Media Port: 12344
Media Proto: RTP/AVP
Media Format: ITU-T G.711 PCMU
Media Format: ITU-T G.723
Media Format: ITU-T G.729
Media Attribute (a): rtpmap:0 PCMU/8000
Media Attribute Fieldname: rtpmap
Media Attribute Value: 0 PCMU/8000
Media Attribute (a): rtpmap:4 G723/8000
Media Attribute Fieldname: rtpmap
Media Attribute Value: 4 G723/8000
Media Attribute (a): rtpmap:18 G729A/8000
Media Attribute Fieldname: rtpmap
Media Attribute Value: 18 G729A/8000
Media Attribute (a): prtime:20
Media Attribute Fieldname: prtime
Media Attribute Value: 20
### B.3.6 SIP - 180 (Ringing) Response

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
<th>Status: 180 Ringing, with session description</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>2006-07-01 14:36:29.145642</td>
<td>172.16.0.12</td>
<td>172.16.0.200</td>
<td>SIP/SDP</td>
<td></td>
</tr>
</tbody>
</table>

**Frame 26 (651 bytes on wire, 651 bytes captured)**

- **Arrival Time:** Jul 1, 2006 14:36:29.145642000
- **Time delta from previous packet:** 0.639062000 seconds
- **Time since reference or first frame:** 28.948160000 seconds
- **Frame Number:** 26
- **Packet Length:** 651 bytes
- **Capture Length:** 651 bytes

**Protocols in frame:** eth:ip:udp:sip:sdp

**Ethernet II**, Src: EdimaTe_54:54:7e (00:0e:2e:54:54:7e), Dst: UniwillC_29:4e:32 (00:03:0d:29:4e:32)

- **Destination:** UniwillC_29:4e:32 (00:03:0d:29:4e:32)
- **Source:** EdimaTe_54:54:7e (00:0e:2e:54:54:7e)
- **Type:** IP (0x0800)

**Internet Protocol**, Src: 172.16.0.12 (172.16.0.12), Dst: 172.16.0.200 (172.16.0.200)

- **Version:** 4
- **Header length:** 20 bytes
- **Differentiated Services Field:** 0x00 (DSJP 0x00: Default; ECN: 0x00)
- **000 00.. = Differentiated Services Codepoint: Default (0x00)**
- **..... ... = ECN-Capable Transport (ECT): 0**
- **..... ...0 = ECN-CE: 0**
- **Total Length:** 657
- **Identification:** 0x0000 (0)
- **Flags:** 0x04 (Don't Fragment)
- **0... = Reserved bit: Not set**
- **1.. = Don't fragment: Set**
- **...0 = More fragments: Not set**
- **Fragment offset:** 0
- **Time to live:** 64
- **Protocol:** UDP (0x11)
- **Header checksum:** 0x6d7b [correct]
- **Source:** 172.16.0.12 (172.16.0.12)
- **Destination:** 172.16.0.200 (172.16.0.200)

**User Datagram Protocol**, Src Port: 5670 (5670), Dst Port: sip (5060)

- **Source port:** 5670 (5670)
- **Destination port:** sip (5060)
- **Length:** 617
- **Checksum:** 0x6741 [correct]

**Session Initiation Protocol**

- **Status-Line:** SIP/2.0 180 Ringing
- **Status-Code:** 180
- **Reason-Phrase:** False

**Message Header**

- **Call-ID:** de6cfc33c1e687ec0ab1949580382f172.16.0.200
- **CSeq:** 1 INVITE
- **From:** "Client" <sip:Client>@tag=12345
- **SIP Display info:** "Client"
- **SIP from address:** sip:Client
- **SIP tag:** 12345
- **To:** "Server" <sip:Server>@tag=4321
- **SIP Display info:** "Server"
- **SIP to address:** sip:Server
- **SIP tag:** 4321

- **Via:** SIP/2.0/UDP 172.16.0.200:5670;branch=z9hG4bKf293a6f85c3f0379140be6abc7c83
- **Contact:** "Server" <sip:172.16.0.12:5070>
- **Domain Binding:** "Server" <sip:172.16.0.12:5070>
- **URL:** "Server" <sip:172.16.0.12:5070>
- **SIP Display info:** "Server"
- **SIP contact address:** sip:172.16.0.12:5070

109
Content-Type: application/sdp
Content-Length: 262

Message body
Session Description Protocol
Session Description Protocol Version (v): 0
Owner/Creator, Session Id (o): 4855 13760799956958020 13760799956958020 IN IP4 172.16.0.200
Owner Username: 4855
Session ID: 13760799956958020
Session Version: 13760799956958020
Owner Network Type: IN
Owner Address Type: IP4
Owner Address: 172.16.0.200
Session Name (s): mysession session
Session Information (i): Subarul.mov
Phone Number (p): +46 8 62018210
Connection Information (c): IN IP4 129.6.55.78
Connection Network Type: IN
Connection Address Type: IP4
Connection Address: 129.6.55.78
Time Description, active time (t): 0 0
Session Start Time: 0
Session Stop Time: 0
Media Description, name and address (m): video 12344 RTP/AVP 0 4 18
Media Type: video
Media Port: 12344
Media Proto: RTP/AVP
Media Format: ITU-T G.711 PCM
Media Format: ITU-T G.723
Media Format: ITU-T G.729
Media Attribute (a): rtpmap:0 PCMU/8000
Media Attribute Fieldname: rtpmap
Media Attribute Value: 0 PCMU/8000
Media Attribute (a): rtpmap:4 G723/8000
Media Attribute Fieldname: rtpmap
Media Attribute Value: 4 G723/8000
Media Attribute (a): rtpmap:18 G729A/8000
Media Attribute Fieldname: rtpmap
Media Attribute Value: 18 G729A/8000
Media Attribute (a): pt ime:20
Media Attribute Fieldname: ptime
Media Attribute Value: 20

110
B.3.7  SIP - 200 (OK) Response

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>2006-07-01</td>
<td>14:36:29.165670</td>
<td>172.16.0.12</td>
<td>172.16.0.200</td>
<td>SIP</td>
<td>Status: 200 OK</td>
</tr>
</tbody>
</table>

Frame 27 (351 bytes on wire, 351 bytes captured)
Arrival Time: Jul 1, 2006 14:36:29.165670000
Time delta from previous packet: 0.0200028000 seconds
Time since reference or first frame: 28.9681880000 seconds
Frame Number: 27
Packet Length: 351 bytes
Capture Length: 351 bytes
Protocols in frame: eth:ip:udp:sip

Ethernet II, Src: EdimaxTe_E5:54:54:7e (00:0e:2e:54:54:7e), Dst: UniviiLC_29:4e:32 (00:03:0d:29:4e:32)
  Destination: UniviiLC_29:4e:32 (00:03:0d:29:4e:32)
  Source: EdimaxTe_E5:54:54:7e (00:0e:2e:54:54:7e)
Type: IP (0x0800)

Internet Protocol, Src: 172.16.0.12 (172.16.0.12), Dst: 172.16.0.200 (172.16.0.200)
  Version: 4
  Header length: 20 bytes
  Differentiated Services Field: 0x00 (DS00 Default; ECN: 0x00)
  0000 00 = Differentiated Services Codepoint: Default (0x00)
  .... 0. = ECN-Capable Transport (ECT): 0
  .... 0. = ECN-CE: 0
  Total Length: 337
  Identification: 0x0000 (0)
  Flags: 0x04 (Don't Fragment)
  0... = Reserved bit: Not set
  .1.. = Don't fragment: Set
  ...0. = More fragments: Not set
  Fragment offset: 0
  Time to live: 64
  Protocol: UDP (0x11)
  Header checksum: 0xe0a7 [correct]
  Source: 172.16.0.12 (172.16.0.12)
  Destination: 172.16.0.200 (172.16.0.200)

User Datagram Protocol, Src Port: 5070 (5070), Dst Port: sip (5060)
  Source port: 5070 (5070)
  Destination port: sip (5060)
  Length: 317
  Checksum: 0x5f27 [correct]

Session Initiation Protocol
  Status-Line: SIP/2.0 200 OK
  Status-Code: 200
  Reason-Phrase: None

Message Header
  Call-ID: deb6cf33c1eb57ec0a19d9e580f68f0e172.16.0.200
  CSeq: 1 INVITE
  From: "Client" <sip:Client>@tag=12345
  SIP Display info: "Client"
  SIP from address: sip:Client
  SIP tag: 12345
  To: "Server" <sip:Server>@tag=4321
  SIP Display info: "Server"
  SIP to address: sip:Server
  SIP tag: 4321
  Via: SIP/2.0/UDP 172.16.0.200:5060;branch=w9h54hKf293a56fb6c3fc0376040bec6abbcc7c93
  Contact: "Server" <sip:172.16.0.12:5070>
  Contact Binding: "Server" <sip:172.16.0.12:5070>
  URI: "Server" <sip:172.16.0.12:5070>
  SIP Display info: "Server"
  SIP contact address: sip:172.16.0.12:5070
  Content-Length: 0

111
B.3.8 SIP - ACK Request

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
<th>Info</th>
</tr>
</thead>
</table>

Frame 28 (318 bytes on wire, 318 bytes captured)

- Arrival Time: Jul 1, 2006 14:36:29.178276000
- Time delta from previous packet: 0.0123456789 seconds
- Time since reference or first frame: 318.987654321 seconds
- Frame Number: 28
- Packet Length: 318 bytes
- Capture Length: 318 bytes
- Protocols in frame: eth/udp/sip

Ethernet II, Src: Univl1C-29:4e:32 (00:03:04:29:4e:32), Dst: EdimaxTe-54:54:7e (00:0e:3e:54:54:7e)
- Destination: EdimaxTe-54:54:7e (00:0e:3e:54:54:7e)
- Source: Univl1C-29:4e:32 (00:03:04:29:4e:32)
- Type: IP (0x0800)

Internet Protocol, Src: 172.16.0.200 (172.16.0.200), Dst: 172.16.0.12 (172.16.0.12)
- Version: 4
- Header length: 20 bytes
- Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
  - 0000 00.. = Differentiated Services Codepoint: Default (0x00)
  - ...0 = ECN-Capable Transport (ECT): 0
  - ......0 = ECN-CE: 0
- Total Length: 304
- Identification: 0x0002 (2)
- Flags: 0x04 (Don't Fragment)
  - 0... = Reserved bit: Not set
  - .1.. = Don't fragment: Set
  - ..0. = More fragments: Not set
- Fragment offset: 0
- Time to live: 64
- Protocol: UDP (0x11)
- Header checksum: 0x0000 (Correct)
- Source: 172.16.0.200 (172.16.0.200)
- Destination: 172.16.0.12 (172.16.0.12)

User Datagram Protocol, Src Port: sip (5060), Dst Port: 5070 (5070)
- Source port: sip (5060)
- Destination port: 5070 (5070)
- Length: 284
- Checksum: 0xa22 (Incorrect, should be 0x71cf)

Session Initiation Protocol
- Request-Line: ACK sip:172.16.0.12:5070 SIP/2.0
- Method: ACK
- Request Packet: False

Message Header
- Call-ID: debdf33c1e6cc67e0a19d9e5801681e172.16.0.200
- CSeq: 1 ACK
- From: "Client" <sip:Client@tag=12345>
  - SIP Display info: "Client"
- To: "Server" <sip:Server@tag=12346>
  - SIP Display info: "Server"
- Via: SIP/2.0/UDP 172.16.0.200:5060;branch=z9hG4bK29adbc3
- Max-Forwards: 70
- Content-Length: 0
B.3.9  SIP - REFER Request

No. Time  Source  Destination  Protocol Info
3988 2006-07-01 15:12:55.549996 172.16.0.200 172.16.0.12  SIP Request: REFER sip:Refer

Frame 3988 (371 bytes on wire, 371 bytes captured)
Arrival Time: Jul 1, 2006 15:12:55.54996000
Time delta from previous packet: 9.639490000 seconds
Time since reference or first frame: 72.301483000 seconds
Frame Number: 3988
Packet Length: 371 bytes
Capture Length: 371 bytes
Protocols in frame: eth:ip:udp:sip

Ethernet II, Src: Unwill1C_29:4e:32 (00:03:0d:29:4e:32), Dest: EdimaxTe_54:54:7e (00:0e:2e:54:54:7e)
Destination: EdimaxTe_54:54:7e (00:0e:2e:54:54:7e)
Source: Unwill1C_29:4e:32 (00:03:0d:29:4e:32)
Type: IP (0x0800)

Internet Protocol, Src: 172.16.0.200 (172.16.0.200), Dest: 172.16.0.12 (172.16.0.12)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
O000 00.. = Differentiated Services Codepoint: Default (0x00)
..... 0... = ECN-Capable Transport (ECT): 0
..... ....0 = ECN-CE: 0
Total Length: 357
Identification: 0x0003 (3)
Flags: 0x04 (Don't Fragment)
0... = Reserved bit: Not set
.1.. = Don't fragment: Set
..0. = More fragments: Not set
Fragment offset: 0
Time to live: 64
Protocol: UDP (11)
Header checksum: 0xe099 [correct]
Source: 172.16.0.200 (172.16.0.200)
Destination: 172.16.0.12 (172.16.0.12)

User Datagram Protocol, Src Port: sip (5060), Dest Port: 5070 (5070)
Source port: sip (5060)
Destination port: 5070 (5070)
Length: 337
Checksum: 0x5a57 [incorrect, should be 0xaddf]

Session Initiation Protocol
Request-Line: REFER sip:Refer SIP/2.0
Method: REFER
Resent Packet: False

Message Header
Call-ID: 5804a78110be0e0d55bdeea3eae4845e172.16.0.200
CSeq: 93809823 REFER
From: "Client" <sip:Client>;tag=12345
SIP Display info: "Client"
SIP from address: sip:Client
SIP tag: 12345
To: "Server" <sip:Server>
SIP Display info: "Server"
SIP to address: sip:Server
Via: SIP/2.0/UDP 172.16.0.200:5060;branch=z9hG4bK2993b49f380f7ef666b394a107fcd3a
Max-Forwards: 70
Refer-To: "172.16.10.10" <sip:Refer>
Content-Length: 0

113
B.3.10  SIP - 202 (Accepted) Response

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol Info</th>
<th>Status: 202 Accepted</th>
</tr>
</thead>
<tbody>
<tr>
<td>3990</td>
<td>2006-07-01 15:12:55</td>
<td>582432</td>
<td>172.16.0.12</td>
<td>172.16.0.200</td>
<td>SIP</td>
</tr>
</tbody>
</table>

Frame 3990 (363 bytes on wire, 363 bytes captured)
Arrival Time: Jul 1, 2006 15:12:55.5824352000
Time delta from previous packet: 0.0242970000 seconds
Time since reference or first frame: 72.3339190000 seconds
Frame Number: 3990
Packet Length: 363 bytes
Capture Length: 363 bytes
Protocols in frame: eth:ip:udp:sip

Ethernet II, Src: EdimaxTE_54:54:7e (00:0e:2e:54:54:7e), Dest: Unwilled_29:4e:32 (00:03:0d:29:4e:32)
Destination: Unwilled_29:4e:32 (00:03:0d:29:4e:32)
Source: EdimaxTE_54:54:7e (00:0e:2e:54:54:7e)
Type: IP (0x0800)

Internet Protocol, Src: 172.16.0.12 (172.16.0.12), Dest: 172.16.0.200 (172.16.0.200)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
0000 00... = Differentiated Services Codepoint: Default (0x00)
...... 0... = ECN-Capable Transport (ECT): 0
...... 0... = ECN-CE: 0
Total Length: 349
Identification: 0x0000 (0)
Flags: 0x04 (Don't Fragment)
0... = Reserved bit: Not set
.1... = Don't fragment: Set
..0... = More fragments: Not set
Fragment offset: 0
Time to live: 64
Protocol: UDP (0x11)
Header checksum: 0xe09b [correct]
Source: 172.16.0.12 (172.16.0.12)
Destination: 172.16.0.200 (172.16.0.200)

User Datagram Protocol, Src Port: 5070 (5070), Dest Port: sip (5060)
Source port: 5070 (5070)
Destination port: sip (5060)
Length: 329
Checksum: 0x1081 [correct]

Session Initiation Protocol
Status-Line: SIP/2.0 202 Accepted
Status Code: 202
Resent Packet: False

Message Header
Call-ID: 5404a781f0be0e0d0a0b6dea3aea48550172.16.0.200
CSeq: 03809823 REFER
From: "Client" <sip:Client>;tag=12345
SIP Display info: "Client"
SIP from address: sip:Client
SIP tag: 12345
To: "Server" <sip:Server>;tag=4321
SIP Display info: "Server"
SIP to address: sip:Server
SIP tag: 4321
Via: SIP/2.0/UDP 172.16.0.200:5060;branch=v9b046k9293b49f380ff7ef66ebe394e107fcd3a
Contact: "Server" <sip:172.16.0.12:5070>
Contact Binding: "Server" <sip:172.16.0.12:5070>
URI: "Server" <sip:172.16.0.12:5070>
SIP Display info: "Server"
SIP contact address: sip:172.16.0.12:5070
Content-Length: 0
Appendix C

Configuration Files

C.1 HIP4BSD - hipconf.xml

<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE hipconf SYSTEM "hipconf.dtd">
<!--
 $Id: hipconf.xml,v 1.8 2006/03/02 14:11:51 jaelen Exp $
 Host Identity Protocol (HIP) configuration file
-->
<hip>
  <chip>
    <!--
    <resolver_socket>tcp:6659</resolver_socket>
    <ctrl_socket>tcp:9966</ctrl_socket>
    -->
    <host localhost="true">
      <key type="private">/etc/hip/hipkey.priv</key>
      <key type="public">/etc/hip/hipkey.pub</key>
      <!-- addr:3ffe::14</addr> Adicionado! DÁ! Erro! -->
      <dgroup>3</dgroup>
      <puzzle>
        <generate interval>38</generate>
        <!-- in format 2"(g-32) -->
        <min_validtime>37</min_validtime>
        <!-- in format 2"(m-32) -->
        <!-- max validtime is:
          generate_interval(2"(g-32)) + min_validtime(2"(m-32)) -->
      </puzzle>
      <ck_bits>10</ck_bits>
      <hip_se>
        <transforms>
          <id>1</id>
          <id>2</id>
        </transforms>
      </hip_se>
      <esp_se>
        <transforms>
          <id>1</id>
          <id>2</id>
        </transforms>
        <!-- Define your own lifetime for SAs
        <lifetime>
        <lt_bytes>0</lt_bytes>
        <lt_add>600</lt_add>
        <lt_use>120</lt_use>
        </lifetime>
        -->
    </host>
  </chip>
</hip>
<!--
</espère>
<!-- Include the addresses of all interfaces on R1
<locator2rl>true</locator2rl>
-->
</host>

<!-- Definição do Host Remoto -->
/host localid="default" localhost="false">
<key type="public">/etc/hip/m1.pub</key>
</host>

</hipd>
</hip>
C.2 OpenHIP - hip.conf

<xml version="1.0" encoding="UTF-8"?>
<hip_configuration>
  <min_lifetime>96</min_lifetime>
  <max_lifetime>255</max_lifetime>
  <reg_type_rve>1</reg_type_rve>
  <lifetime>217</lifetime>
  <reg_type>1</reg_type>
  <cookie_difficulty>10</cookie_difficulty>
  <packet_timeout>10</packet_timeout>
  <max_retries>5</max_retries>
  <sa_lifetime>900</sa_lifetime>
  <preferred_hi>lap-1024</preferred_hi>
  <send_hi_name>yess</send_hi_name>
  <dh_group>3</dh_group>
  <db_lifetime>900</db_lifetime>
  <rl_lifetime>300</rl_lifetime>
  <failure_timeout>50</failure_timeout>
  <maxl>5</maxl>
  <aul>600</aul>
</hip_configuration>

<transforms>
  <id>1</id>
  <id>2</id>
  <id>3</id>
  <id>4</id>
  <id>5</id>
  <id>6</id>
</transforms>

</hip_configuration>

<esp_sa>
  <transforms>
    <id>1</id>
    <id>2</id>
    <id>3</id>
    <id>4</id>
    <id>5</id>
    <id>6</id>
  </transforms>
</esp_sa>

<dns_server>130.192.86.29</dns_server>
<disable_dns_lookups>yes</disable_dns_lookups>
<disable_notif>no</disable_notif>
<disable_dns_thread>not</disable_dns_thread>
<min_lifetime>96</min_lifetime>
<max_lifetime>255</max_lifetime>
<reg_type_rve>1</reg_type_rve>
<lifetime>255</lifetime>
<reg_type>1</reg_type>
</hip_configuration>
The parameters are defined as follows:

- **cookie_difficulty** - sets the puzzle difficulty for outgoing puzzles. The difficulty is the puzzle value $K$ indicating the
- number of lowest order bits that must be zero after the Responder hashes a random number plus other puzzle data.
- **cookie_lifetime** - sets the puzzle's valid lifetime, $2^{(lifetime - 32)}$ seconds; the default value of $39 = 2^{(39 - 32)} = 128$ seconds.
- **packet_timeout** - seconds to wait before retransmitting a HIP packet
- **max_retries** - the number of packet retransmissions before failure
- **sa_lifetime** - seconds before an SA expires and must be updated; for Linux kernel-mode IPsec, the lifetime is set when adding the SA and the kernel will expire the SA automatically
- **preferred_hi** - sets the preferred Host Identity to use from my_host_identities.xml
- **send_hi_name** - whether or not to include the Domain Identifier along with the Host Identity in HIP control packets
- **dh_group** - sets the default Diffie Hellman group ID to use; possible values 1-6 are defined in the HIP specification;
- **default value of 3** specifies the 1536-bit MODP group.
- **dh_lifetime** - number of seconds to cache precomputed Diffie Hellman values
- **r1_lifetime** - number of seconds to cache precomputed R1 packets
- **failure_timeout** - number of seconds to remain in the state E_FAILED before transitioning to UNASSOCIATED, and how long to
- remain in the state REKEYING before timing out.
- **msi** - maximum segment lifetime - the maximum time in seconds that a TCP segment is expected to spend in the network, used to time out states
- **ual** - unused association lifetime - number of seconds after which an active association is brought down due to inactivity (no packets sent or received)
- **hip_sa and esp_sa transforms** - specifies the HIP/ESP cryptographic transforms to advertise in the R1 packet; the order they are listed determines the order inserted into the R1 packet; numeric values correspond to Suite-IDs listed in the HIP specification - e.g., 1 represents AES-CBC with HMAC-SHA1
- **dht_server** - IP address specifying a Bamboo DHT server for storing/retrieving HIP information; this is optional - if no server is specified, the DHT will not be used
- **dht_server_port** - XML RPC port for the DHT server
- **dns_server** - IP address specifying the DNS server that stores the HIP RRAs; this is optional - if no server is specified, the system's default DNS server is contacted (as specified in /etc/resolv.conf for Linux or in the Window's registry)
- **disable_dns_lookups** - disable address resolution for peers whose IP address is unknown
- **disable_notify** - disable the sending of informational NOTIFY packets
- **disable_dns_thread** - disable the dummy DNS server that listens on a loopback address for requests ending with "hip"; this will prevent HIP from interfering with the system's normal address resolution
- **min_lifetime** - Rendezvous server parameter: minimum registration lifetime offered by the RVS - $2^{(min_lf - 64)}$ seconds, default is 96
- **max_lifetime** - Rendezvous server parameter: minimum registration lifetime offered by the RVS - $2^{(max_lf - 64)}$ seconds, default is 255
- **reg_type_rvs** - Rendezvous server parameter: registration type offered by the RVS, default is 1
- **lifetime** - Lifetime that the client requests during registration, default is 255
- **reg_type** - registration type requested by the client during registration, default is 1

118
C.3 OpenHIP - my_host_identities

```xml
<xml version="1.0" encoding="UTF-8"/>
<my_host_identities/>
</xml>```

C.4 OpenHIP - known_host_identities

```xml
<xml version="1.0" encoding="UTF-8"/>
<known_host_identities/>
</xml>
```

119
Appendix D

How-Tos

D.1 Installation of the JMF on Linux

The installation of the JMF in a Linux platform requires some dependencies to be satisfied. We'll explain the procedure step-by-step.

$J2RE - JRE's installation directory
$JMF - JMF's installation directory

1. Copy the file $JMF/lib/jmf.properties to $J2RE/lib/jmf.properties
2. Copy all the .jar files from the directory $JMF/lib to $J2RE/lib/ext/
3. Copy all the .so files from $JMF/lib to $J2RE/lib/i386/

The installation can be verified with the JMF diagnostics applet at http://java.sun.com/products/java-media/jmf/2.1.1/jmfdiagnostics.html.

The applet must show the following:

JMF Version ... 2.2.1e
All Java Build
Native Libraries Found

JMF Version can show a different version, obviously.

For the stream to work properly, it's necessary that the file /etc/hosts in the machine that initiates the streaming - the server machine in our case - presents the following structure.

127.0.0.1 localhost
xxx.xxx.xxx.xxx HostName

Where xxx.xxx.xxx.xxx represents the address of the interface on the local machine. HostName is the local hostname. Something like

127.0.0.1 localhost HostName

will cause the stream not to work.

After this alteration to the /etc/hosts file, it's necessary to restart networking with

/etc/init.d/network restart
D.2 Installation of the HIP4BSD on Linux

HIP4BSD HowTo [Linux]

Source Roadmap:

src/hip.................source code for HIP user level programs
conf/.....................Example configurations (HIP and Linux kernel)
linux-2.6..............Linux 2.6.18.2 with some modification (see below)

Building:

Follow the kernel build instruction found in linux-2.6/README
When configuring the kernel, enable the following options:

Networking -> Networking options -> PF_KEY sockets
Networking -> Networking options -> XFRM Identity extension
Networking -> Networking options ->
    XFRM SA first usage notification Networking -> Networking options -> The IPv6 Protocol
Networking -> Networking options -> Socket level support for HIP
Device Drivers -> Network device support -> Dummy net driver support
Cryptographic options -> HMAC support
Cryptographic options -> SHA1 digest algorithm
Cryptographic options -> AES cipher algorithms

There's a linux config example on hip4bsd/conf/linux.config

Navigate to hip4bsd/src/hip and there you should make

./configure
make
make install

Some error might occur. If so, they should be easily
corrected by adding "-ldl" to the "LIBS" var in the following directories' Makefiles:

/hip/hip4bsd/src/hip
/hip/hip4bsd/src/hip/hipctrl
/hip/hip4bsd/src/hip/hipd
/hip/hip4bsd/src/hip/hip-keygen
/hip/hip4bsd/src/hip/libpfkey

Generate keypair:

In hip directory:

cd hip-keygen

To view the command line options:

./hip-keygen

To create a new RSA key:

./hip-keygen -r

A copy the resulting keys should appear in the same directory, and should be named HIT.priv and HIT.pub.
Copy these to (for example) /etc/hip/. Also, we recommend that you create symbolic links hipkey.(priv|pub) in the same directory, for easier configuration.

Configuration of the HIP daemon:

Create a configuration file for HIP daemon, and save it as /etc/hipconf.xml
An example configuration file can be found in the "main" source directory or you can look into our examples.

Few options are described here:

```<ctrl_socket> Enable if you need to control HIP daemon from programs
<host localhost="true"> Local host setup
<key> Paths to local and private keys
<k_bits> The difficulty factor of "anti-dos" cookie
<lifetime> Default lifetimes of the security associations
<extid> Identifier for better control from programs
<p localhost="false"> Static peer host entry. Required tags: key, addr
```

For configuring the peer you should exchange between modes the file named HIT.pub. Create a symbolic link to it called "hip-node2.pub" Then the following lines should be added to the hipconf.xml files:

```<host localhost="default" localhost="false">
<key type="public"> /etc/hip/hip-node2.pub </key>
</host>
```

Starting HIP:

You must be superuser to start HIP daemon.
The applications utilizing HIP, can be ran with normal user privileges.
Before starting HIP daemon, make sure that you have enabled dummy0 interface.
This can be done:

```ifconfig dummy0 up```

In hip directory:

```cd hipd```

To get all debugging information:

```./hipd -nt 65535```

To get command line information:

```./hipd -h```

We suggest running with all debugging information, since the program is experimental, and the problems are easier to solve, if you provide full debug information in case the program misbehaves.

You will need two computers to run HIP successfully.
To just send I1 only one computer is required.

You will notice that the key has been registered by the HIPD when it outputs during the startup something like:

```
Garbage: [08:45:06] [hipd_context_get_by_bits]:
Local HIT: 112d:5d76:a3c1:7281:5b9c:f798:3b7d:8835
Garbage: [08:45:06] [hipd_context_get_by_bits]:
Remote HIT: 1173:2be2:259:be2:7c2a:3295:5e02:410f
Garbage: [08:45:06] [hipd_context_get_by_bits]:
Searching context for key: 9923e847
Garbage: [08:45:06] [hipd_context_get_by_bits]:
Found state for key by hit: 9923e847
```

122
Naturally your HITs and keys will vary.

The daemon will configure the dummy0 interface with an IPv6 address which is the same as his HIT. The daemon will also create a SPD in each peer, but not SSKs.

./setkey -Dp will show something like:

```
11a9:ac04:ff2e:64cb:d251:1c33:c43:59c5[any] any
Policy:[invalid ipsec mode]
created: Mar 28 16:48:59 2006 lastused:
lifetime: 0(s) validtime: 0(s)
spid=153 seq=0 pid=17185
refcnt=
```

In this example

```
1118:85bb:b590:4c6:ab86:acce:c3b4:1212 is the HIT of the localhost and
11a9:ac04:ff2e:64cb:d251:1c33:c43:59c5 is the HIT of the remote HIP peer.
```

The first application data packet, sent to a HIT address, will cause HIP base exchange to be run. You can try to do something like:

```
ping6 11a9:ac04:ff2e:64cb:d251:1c33:c43:59c5
```

which will trigger the base exchange.

The 4 base exchange packets can be captured using tcpdump -ni eth0 'proto 253'

(Etheral patched for the OpenHIP implementation will not recognize these packets as HIP. They will appear as unknown IPv6 with protocol number 0xFD = 253d)

After the base exchange is complete you should see the ping6 encapsulated (ESP) packets being exchanged.

There will be created two SSKs between the peers.

./setkey -D shows something like:

```
esp mode=3 spi=2847002560(0x22d9d160) reqid=0(0x00000000)
E: aes-cbc fd413f63 33e65e44 061036f6 04ebc26
A: hmac-sha1 56d5df26c 26219977 73db4e6a 394ca4f4 42313d3e
seq=0x00000000 replay=0 flags=0x00000000 state=mature
created: Mar 28 17:00:31 2006 current: Mar 28 17:00:46 2006
diff: 15(s) hard: 900(s) soft: 720(s)
last: Mar 28 17:00:32 2006 hard: 0(s) soft: 0(s)
current: 576(bytes) hard: 0(bytes) soft: 0(bytes)
allocated: 9 hard: 0 soft: 0
```

```
esp mode=3 spi=5847002560(0x22d9d160) reqid=0(0x00000000)
E: aes-cbc 2b63fffcd1069a2 dda54a4 2e4825d6
A: hmac-sha1 89c5e14071995e97 6a8e6d41 96c1cbf0 045e2c5f
seq=0x00000000 replay=0 flags=0x00000000 state=mature
created: Mar 28 17:00:31 2006 current: Mar 28 17:00:46 2006
diff: 15(s) hard: 0(s) soft: 0(s)
last: Mar 28 17:00:31 2006 hard: 0(s) soft: 0(s)
current: 1560(bytes) hard: 0(bytes) soft: 0(bytes)
allocated: 10 hard: 0 soft: 0
```

physical interfaces of the peering nodes.
D.3 Installation of the HIPL on Linux

HIPL HowTo

Source Roadmap:

doc/.................Documentation about HIPL
patches/..............Some patches needed during installation

Building:

During the installation you’ll need to apply some patches to
the kernel source. These patches are available in the patches/ folder mentioned before.

Best Patch:
The BEET patch consist in two incremental patches.
The first patch to be applied is the simple
one and the second one is the interfamily.
Therefore if you would like to patch Linux kernel
2.6.13.1 you first apply beet-patch-2.6.13.1 and then interfamily-2.6.13.1
The main difference between these two patches is that the former
supports the same families between inner and outer addresses
(as a result the only allowed cases are inner-outer=IPv4 and inner-outer=IPv6),
whereas the latter supports also the cross-family transformation.
The reason why we created incremental patches is due to requirements from Linux community.

Download the latest hipl e.g. from the nightly tarball:

Then, do as follows:

cd /where/you/extracted/the/linux-2.6.13.1
patch -p1 < /where/you/extracted/hipl--main--2.6/patches/beet/beet-patch-2.6.13.1
patch -p1 < /where/you/extracted/hipl--main--2.6/patches/beet/interfamily-2.6.13.1
patch -p1 < /where/you/extracted/hipl--main--2.6/patches/kernel/policy-sleep-2.6.13.1-v5.patch
patch -p1 < /where/you/extracted/hipl--main--2.6/patches/kernel/ipv6_addr_del.patch.2.6.14.4.slokov

Then, recompile the kernel following the building
instructions found at http://infrahipl.hiiit.fi/hipl/manual/ch04.html

After you have successfully compiled and installed the
HIP kernel and rebooted the host, you need to compile the
userspace applications in order to use HIP. Start by moving to the top level directory of HIPL:

cd hipl

Run autogen.sh
.

Next, build the hipd, libnet6, tools and test directory as follows:

./configure
make
make install

For future RVS support you should make ./configure --enable-rvs
Other options are available, but some compilation errors appeared when trying:

./configure --enable-rvs --enable-hi3

and
./configure --enable-rvs--enable-openHDT

As this isn't needed right now, we just enabled the RVS support.

Some other errors might occur (as in HIP4BSD).
If so, they should be easily corrected by adding "-ldl"
to the "LIBS" var in the following directories' Makefiles:

  hipl--main--2.6/
  hipl--main--2.6/tools
  hipl--main--2.6/hipd
  hipl--main--2.6/test

It is not necessary to "make install" the applications.
You can execute them straight from their source directories.
Note that making from the top directory does not currently build e.g. java libraries or telnet.
Please refer to the installation manual at http://infradhip.hitl.fi/hipl/manual/index.html

Generating keypairs:
Using tools/hipconf you should generate 8 different files

  hip_host_dsa_key_anon
  hip_host_dsa_key_pub
  hip_host_rsa_key_anon
  hip_host_rsa_key_pub
  hip_host_dsa_key_anon.pub
  hip_host_dsa_key_pub.pub
  hip_host_rsa_key_anon.pub
  hip_host_rsa_key_pub.pub
  HIPL HovTo

Source Roadmap:

doc/.................Documentation about HIPL
patches/.............Some patches needed during installation

Building:

During the installation you'll need to apply some patches
to the kernel source. These patches are available in the patches/ folder mentioned before.

Bet Patch:
The BEET patch consist in two incremental patches.
The first patch to be applied is the simple one and the second one is the interfamily.
Therefore if you would like to patch Linux kernel 2.6.13.1 you first
apply beet-patch-2.6.13.1 and then interfamily-2.6.13.1
The main difference between these two patches is that the
former supports the same families between inner and outer
addresses (as a result the only allowed cases are inner=outer=IPv4
and inner=outer=IPv6), whereas the latter supports also the cross-family transformation.
The reason why we created incremental patches is due to requirements from Linux community.

Download the latest hipl e.g. from the nightly tarball:

Then, do as follows:

cd /where/you/extracted/the/linux-2.6.13.1
patch -p1 < /where/you/extracted/hipl--main--2.6/patches/beet/beet-patch-2.6.13.1
patch -p1 < /where/you/extracted/hipl--main--2.6/patches/beet/interfamily-2.6.13.1
patch -p1 < /where/you/extracted/hipl--main--2.6/patches/kernel/policy-sleep-2.6.13.1-v5.patch
patch -pl < /where/you/extracted/hipl--main--2.6/patches/kernel/ipv6_addr_del.patch.2.6.14.4.slavov

Then, recompile the kernel following the building instructions found at

After you have successfully compiled and
installed the HIP kernel and rebooted the host,
you need to compile the userspace applications in order
to use HIP. Start by moving to the top level directory of HIPL:

cd hipl

Run autogen.sh

./autogen.sh

Next, build the hipd, libinet6, tools and test directory as follows:

./configure
make
make install

For future RVS support you should make ./configure --enable-rvs
Other options are available, but some compilation errors appeared when trying:

./configure --enable-rvs --enable-ht3

and

./configure --enable-rvs--enable-opendht

As this isn't needed right now, we just enabled the RVS support.

Some other errors might occur (as in HIP4BSD).
If so, they should be easily corrected by adding "-ldl" to the "LIBS"
var in the following directories' Makefiles:

hipl--main--2.6/
hipl--main--2.6/tools
hipl--main--2.6/hipd
hipl--main--2.6/test

It is not necessary to "make install" the applications.
You can execute them straight from their source directories.
Note that making from the top directory
does not currently build e.g. java libraries or telnet.
Please refer to the installation manual

Generating keypairs:

Using tools/hipconf you should generate 8 different files

hip_host_dsa_key_anon
hip_host_dsa_key_pub
hip_host_rsa_key_anon
hip_host_rsa_key_pub
hip_host_dsa_key_anon.pub
hip_host_dsa_key_pub.pub
hip_host_rsa_key_anon.pub
hip_host_rsa_key_pub.pub
These files should be copied to /etc/hip/
You can find our generated files here

The specifications recommend that a host should have
two public and two anonymous host identifiers.
The prefix of all keys is "hip_host". After that is the algo "dsa" or "rsa".
Then we have the privacy type of the hit, either "public"
(can be e.g. published in dns) or "anon" (can be changed frequently).
Now, after this there can be .pub suffix. If it is not present,
it means it's the private key. If it is present,
it means it's the public key part of the private key.

Configurating and testing HIPL:

There are 2 ways to establish communication between the 2 HIP hosts.

Using HIP aware applications:

The mapping between HIP and IPv6 is accomplished using the
/etc/hosts and /etc/hip/hosts (which needs to be created) files.
In the /etc/hosts you have to append an entry mapping the
IPv6 of the peer's physical interface with the peer's Hostname

# IPv6 - HOSTNAME Mapping

In the /etc/hip/hosts file, you have to make the mapping between the HIT and the peer's Hostname

# HIT - HOSTNAME Mapping
1137:039f:79bc:024:ff2f:59fc:2f07:5dc6 HIPHostM1

The HIT used is one of the HITs automatically configured to the dummy device.
The dummy device is only up when the daemon is running.
The use of one of the 4 HITs attributed to the dummy will
determine the combination of the RSA/DSA and pub/anon used (we think).
An example of the /etc/hip/hosts file can be found here.

This configuration can be tested using the test tool
supplied and as explained at http://infrahip.hiit.fi/hip/ manual/ch06.html
test/client-test-client-gui is actually a HIP unaware application,
but is is configured to resolve the HIT using the two files previously mentioned.

Using HIP unaware applications:

The mapping between HIP and IPv6 is accomplished using the hipconf tool:
tools/hipconf add map hit ipv6

This tool must be used when the hipd/hipd daemon is running.
The mapping will be automatically be deleted when the daemon is stopped.

When the hipconf tool is used 2 SPDs will be created. These SPDs use the HITs of both HIP Hosts.

setkey -DP will show something like:

created: Apr 5 11:57:14 2006 lastused:
lifetime: 0(a) validtime: 0(a)
spid=8 seq=1 pid=9756
refcnt=1
The SAs will be formed, as usual, after the first data packet is exchanged.
You can try ping6 116b:3e29:5df0:30bb:a9e2:7219:138:0c6b

setkey -D will show something like:

```
wp mode=3 spid=1904426852(0x71833f64) reqid=0(0x00000000)
E: asec-cbc b08f3b25 f416ec12 20335c78 98db49f6
A: hmac-sha1 adda3529 c61fa979 7a4c5e72 93c37f62 e07b9f3f
seq=0x00000000 replay=0 flags=0x00000000 state=mature
created: Apr 5 12:01:51 2006 current: Apr 5 12:02:00 2006
diff: 9(s) hard: 0(s) soft: 0(s)
last: Apr 5 12:01:51 2006 hard: 0(s) soft: 0(s)
current: 78000000000 hard: 0(bytes) soft: 0(bytes)
allocated: 5 hard: 0 soft: 0
sadb_seq=1 pid=9822 refcnt=0
```

Where 2001:680:2380:777a:2c0:dfff:fe29:4e32 and

Early testing:

The specifications recommend that a host should have two
public and two anonymous host identifiers.
The prefix of all keys is "hip:host".
After that is the algo "dnc" or "raa". Then we have the privacy type of the hit,
either "pub" (can be e.g. published in dns) or "anon" (can be changed frequently). Now, after this there can be .pub suffix.
If it is not present, it means it's the private key. If it is present, it means it's the public key part of the private key.

Configuring and testing HIP:

There are 2 ways to establish communication between the 2 HIP hosts.

Using HIP aware applications:

The mapping between HIP and IPv6 is accomplished using the
/etc/hosts and /etc/hip/hosts (which needs to be created) files.
In the /etc/hosts you have to append an entry mapping the
IPv6 of the peer's physical interface with the peer's Hostname

```bash
# IPv6 - HOSTNAME Mapping
```

129
In the /etc/hip/hosts file, you have to make the mapping between the HIT and the peer's Hostname.

```
# HIT - HOSTNAME Mapping
1137:8397:75bc:024:ff26:f9fe:2f07:5dc6 HIPHostM1
```

The HIT used is one of the HITs automatically configured to the dummy device. The dummy device is only up when the daemon is running. The use of one of the 4 HITs attributed to the dummy will determine the combination of the RSA/DSA and pub/anon used. An example of the /etc/hip/hosts file can be found here.

This configuration can be tested using the test tool supplied and as explained at http://infrahip.hit.pl/hipl/manual/ch06.html test/ctxbpt-client-gai is actually a HIP unaware application, but is configured to resolve the HIT using the two files previously mentioned.

Using HIP unaware applications:

The mapping between HIP and IPv6 is accomplished using the hipconf tool:
```
tools/hipconf add map hit ipv6
```

This tool must be used when the hipd/hipd daemon is running. The mapping will be automatically be deleted when the daemon is stopped.

When the hipconf tool is used 2 SPDs will be created. These SPDs use the HITs of both HIP Hosts.

setkey -DP will show something like:
```
116b:3e29:5df0:30bb:a9e2:7219:138:c685
Policy: [Invalid ipsec protocol]
created: Apr 5 11:57:14 2006 last used:
lifetime: 0(s) validtime: 0(s)
spid=8 seq=1 pid=9766
refcnt=1
```

```
116b:3e29:5df0:30bb:a9e2:7219:138:c685
Policy: [Invalid ipsec protocol]
created: Apr 5 11:57:14 2006 last used:
lifetime: 0(s) validtime: 0(s)
spid=0 seq=0 pid=9766
refcnt=1
```

The SPDs will be formed, as usual, after the first data packet is exchanged. You can try ping6 116b:3e29:5df0:30bb:a9e2:7219:138:c685

setkey -D will show something like:
```
esp mode=3 spid=1904268520x1853f64) reqid=0x00000000
E: seq 0x188fc126238c78 ef6e49f8
A: hmac-sha1 add3239 d6f1f979 76cc3762 07b5b03f
seq=0x00000000 replay=0x00000000 state=nature
created: Apr 5 12:01:51 2006 current: Apr 5 12:02:00 2006
diff: 0(s) hard: 0(s) soft: 0(s)
last: Apr 5 12:01:51 2006 hard: 0(s) soft: 0(s)
```

130
current: 780(bytes) hard: 0(bytes) soft: 0(bytes)
allocated: 5 hard: 0 soft: 0
sadb_seq=1 pid=9822 rfcnt=0

esp mode=3 spi=ff84515850(0x469789ca) reqid=0(0x00000000)
E: asec-cbc 3363da8 d0c5a92c f68df47f 73bf77a
A: hmac-sha1 b956f24 9ef8fd72 c0e0c8f8 ba55b42 8e5ec8d4
seq=0x00000000 replay=0 flags=0x00000000 state=mature
created: Apr 5 12:01:51 2006 current: Apr 5 12:02:00 2006
diff: 0(s) hard: 0(s) soft: 0(s)
last: Apr 5 12:01:51 2006 hard: 0(s) soft: 0(s)
current: 320(bytes) hard: 0(bytes) soft: 0(bytes)
allocated: 5 hard: 0 soft: 0
sadb_seq=0 pid=9822 rfcnt=0

2001:690:2380:777a:2c0:diff:fe6:ec45 are the IPV6 addresses of the physical interfaces.
Bibliography

November 2004.
2003.
February 2006.
[22] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and  
structure. 2002.